ISSN 0265-2919

80p 49

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO

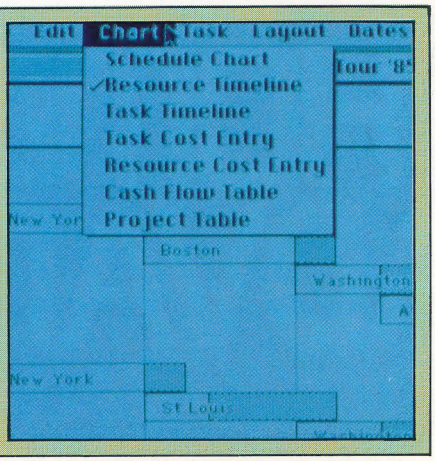
An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

Next Week

- **We begin a new tutorial course on PASCAL, a highly structured language that has had many of its features adopted by other computer languages.**
- **Our series on computer communications comes to its conclusion with an examination of bulletin boards.**
- **Mac Project is a program that enables you to plan a task in the most efficient manner. We take a critical look at the package.**



QUIZ

1) Can you suggest a reason why the polyphonic mode and the rhythm section can't be played simultaneously on the Music Maker?
2) Microspeech claim that My Talking Computer has 128 Kbytes of memory. Is this claim accurate?
3) When sending a message to a second computer on a local area network, which system does not require the data to pass through a third computer?
4) What is a 'parameter block'?

Answers To Last Week's Quiz

1) The width of the sensors themselves will limit the accuracy of the measurement.
2) No. Because the Human Edge has only the subjective data provided by the user to work on, it cannot give an objective analysis.
3) The effect of the * is to bypass the BASIC interpreter and direct the command to the operating system.
4) A silicon disk is an area of RAM that is treated by the computer as an external floppy disk.

Editor Mike Wesley; **Art Editor** Claudia Zeff; **Production Editor** Bobby Pickering; **Technical Editors** Steve Colwill, Brian Morris; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Staff Writer** Stephen Malone; **Sub Editor** Jonathan Kaye; **Contributors** Geoff Bains, Harvey Mellar, Joe Pritchard, Steve Malone, Karl Dallas, Surya, Steve Colwill; **Software Consultants** Pilot Software City; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Maurice Geller; **Production Manager** Peter Taylor-Medhurst; **Subscription Manager** Christine Allen; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 14 Rathbone Place, London W1P 1DE; **© APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Heanor Gate Printing Ltd, Derby**

HOW TO OBTAIN ISSUES AND BINDERS FOR THE HOME COMPUTER ADVANCED COURSE – Issues can be obtained by placing an order with your newsagent or direct from our subscription department. If you have any difficulty obtaining any back issues from your newsagent, please write to us stating the issue(s) required and enclosing a cheque for the cover price of the issue(s). **AUSTRALIA** – please write to: Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G, Melbourne, Victoria 3001. **MALTA, NEW ZEALAND & SOUTH AFRICA** – Back numbers are available at cover price from your newsagent. In case of difficulty, write to the address given for binders.
UK/EIRE – Price: 80p/IR£1. Subscription: 6 months: £23.92. 1 Year: £47.84. Binder: please send £3.95 per binder, or take advantage of our special offer in early issues. **EUROPE** – Price: 80p. Subscription: 6 months air: £43.68. Surface: £34.84. 1 year air: £87.36. Surface: £69.68. Binder: £5.00. Airmail: £8.25. **MALTA** – Obtain binders from your newsagent or Miller (Malta) Ltd, MA Vassalli Street, Valetta, Malta. Price: £3.95. **MIDDLE EAST** – Price: 80p. Subscription: 6 months air: £45.76. Surface: £34.84. 1 year air: £91.52. Surface: £69.68. Binder: £5.00. Airmail: £8.25. **AMERICAS/ASIA/AFRICA** – Price: US/CAN$1.95/80p. Subscription: 6 months air: £54.08. Surface: £34.84. 1 year air: £108.16. Surface: £69.68. Binder: £5.00. Airmail: £9.50. **SOUTH AFRICA** – Price: SA R1.95. Obtain binders from any branch of Central News Agency or Intermag, PO Box 57394, Springfield 2137. **SINGAPORE** – Price: Sing $4.50. Obtain binders from MPH Distributors, 601 Sims Drive, 03-07-21, Singapore 1438. **AUSTRALASIA/FAR EAST** – Price: 80p. Subscription: 6 months air: £58.24. Surface: £34.84. 1 year air £116.48. Surface: £69.68. Binder: £5.00. Airmail: £9.75. **AUSTRALIA** – Price: Aus$1.95. Obtain binders from First Post Pty Ltd, 23 Chandos Street, St Leonards, NSW 2065. **NEW ZEALAND** – Price: NZ$2.25. Obtain binders from your newsagent or Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington.
ADDRESS FOR BINDERS AND BACK ISSUES – Orbis Publishing Limited, Orbis House, Bedfordbury, London WC2 4BT. Telephone 01-379 5211. Cheques/postal orders should be made payable to Orbis Publishing Limited. Binder prices include postage and packing and prices are in stirling. Back issues are sold at the cover price, and we do not charge carriage in the UK.
NOTE – Binders and back issues are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK and Australian markets only. Binders and Issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.
ADDRESS FOR SUBSCRIPTIONS – Orbis Publishing Limited, Hurst Farm, Baydon Road, Lambourn Woodlands, Newbury Berks, RG16 7TW. Telephone: 0488-72666. All cheques/postal orders should be made payable to Orbis Publishing Limited, Postage and packaging is included in subscription rates, and prices are given in sterling.

COVER PHOTOGRAPHY BY MARCUS WILSON-SMITH SHOT AT CAFÉ PELICAN, ST MARTIN'S LANE, WC2

KEY NOTES

The Commodore Music Maker is an ingeniously simple package based around a two-octave piano keyboard that clips onto the Commodore 64 to give the micro a musical touch. What kind of effects can be achieved with this well-priced combination of hardware and menu-driven software?

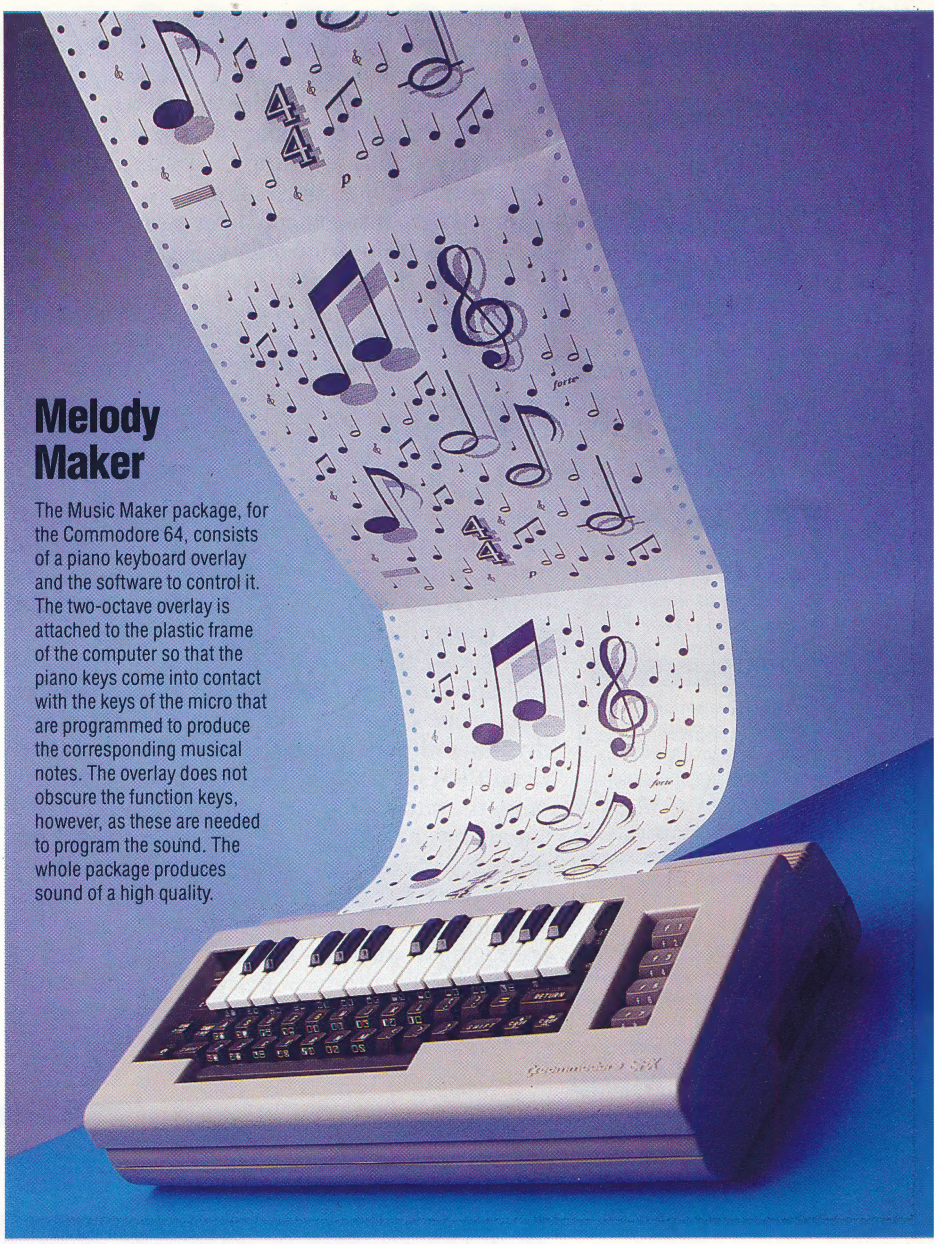
Certain facilities on home computers positively invite the development of peripherals to extend them to their full potential. This is particularly true of their sound and graphics capabilities, for which an enormous range of add-on products has been developed. Such peripherals are especially desirable for the Commodore 64, which lacks even the simple sound commands available on most other micros. A wealth of music making packages have become available for this micro, but whatever their merits, tunes still had to be played on the typewriter keyboard and not on the piano-style keyboard that musicians are familiar with. However, with the introduction of the Music Maker, Commodore has produced a package that provides direct access to the micro's highly rated SID (Sound Interface Device) chip and a keyboard overlay that allows the player to use the familiar black-and-white piano keyboard.

The overlay, constructed in solid plastic, fits snugly over the body of the 64's keyboard around the typewriter and function keys. The Music Maker keyboard, spanning two octaves, is made of a softer plastic and each key is individually attached to the overlay. There are teeth-like projections on the underside of the keys so that, when pressed, a key will push down a specific typewriter key underneath that has been programmed to produce a certain note.

For such a cheap and simple method the system works surprisingly well. The solid plastic of the overlay holds the piano keys firmly in place, and even while performing fast runs up and down the keyboard, notes are seldom missed.

MUSIC MAKER SOFTWARE

The software provided with the Music Maker package is supplied on either cassette or disk. The program is menu-driven, mostly using the eight function keys. These enable the user to alter the sound — or number of sounds — produced from the piano keyboard, by changing either the shape of the sound wave or the pitch. Facilities are also provided for a percussion and bass rhythm to be used as an accompaniment, and a sequencer allows tunes to be programmed and played back. Programmed sounds and sequences can be SAVEd or LOADed from cassette or disk.



Melody Maker

The Music Maker package, for the Commodore 64, consists of a piano keyboard overlay and the software to control it. The two-octave overlay is attached to the plastic frame of the computer so that the piano keys come into contact with the keys of the micro that are programmed to produce the corresponding musical notes. The overlay does not obscure the function keys, however, as these are needed to program the sound. The whole package produces sound of a high quality.

IAN McKINNELL

Each of the eight voices can be programmed to produce a different sound. Using the F6 — modify voice — option, the user can select which voice is to be altered. Then a series of options appears on the screen, beginning with Attack, Decay, Sustain and Release, a number in the range 0 to 15 being chosen for each of these parameters. The programmer is then asked whether the filters are to be implemented. (These cut out sounds above and below specified frequencies.) If you answer yes, another series of questions appears to determine the frequency and level of the filter.

The sequencer is an important component of synthesised music that is curiously overlooked in many microcomputer music packages. Programmed in two parts, it plays a repeated sequence. First, the programmer enters the notes of the sequence via the piano keyboard, then the length of each note — and hence the rhythm of the tune — is tapped in using the F5 key.

MUSIC MAKER

PRICE

£29.95

SOFTWARE

The software provided with the package is available on either disk or cassette

DOCUMENTATION

Music Maker is supplied with two manuals. The user guide tells you everything you may need to know to get started, although more experienced users may find that it lacks the detailed information that they may need. A second manual, called 'Start Playing Keyboard', gives the score of a number of popular tunes

STRENGTHS

At the price, Music Maker is a very worthwhile package, which allows the Commodore 64 to be used as a real musical instrument

WEAKNESSES

The package is severely restricted in its range of options, particularly with regard to the preprogrammed drum and bass rhythms

The three percussion rhythms provided are limited but adequate. Bass rhythms follow the same pattern as the percussion and three options allow the programmer to turn the bass on or off or vary the pitch. Depending on which rhythm is being played, this will drop the pitch of the bass line either by a fifth or an entire octave. The speed of the rhythm can be altered by pressing the cursor keys — Down slows the tempo, Right increases it.

However, many of the options cannot be used together. Although bass and percussion can be played simultaneously, with the user laying a tune on the keyboard over the top, this can only be done while the keyboard is in monophonic mode. It is therefore impossible to play chords with the rhythm in the background.

This is also true for the sequencer option. Many pop groups today dispense with the conventional rhythm section of bass and drums and perform with a sequencer backing instead. There is no facility on the Music Maker to do this, however, or even to play the sequencer with the rhythm section provided. This restriction is actually imposed by the hardware. Because the SID chip has only three voices, one cannot expect a bass and percussion rhythm and polyphonic sound at the same time. Even the Glissando (an option which gradually raises the pitch of the note while the key is pressed together with the Space bar) will not work with the rhythm section, although this problem probably results from the fact that the sound is generated digitally. To keep bass and drums going as well as producing a small enough change in the pitch of the sound to produce a smooth glide seems beyond the capabilities of the processor.

It is more difficult to understand why the software writers chose not to allow users to program their own bass and percussion rhythms into the machine. The techniques would be the same as used in the sequencer, and a programmable rhythm section would have greatly added to the versatility of the package.

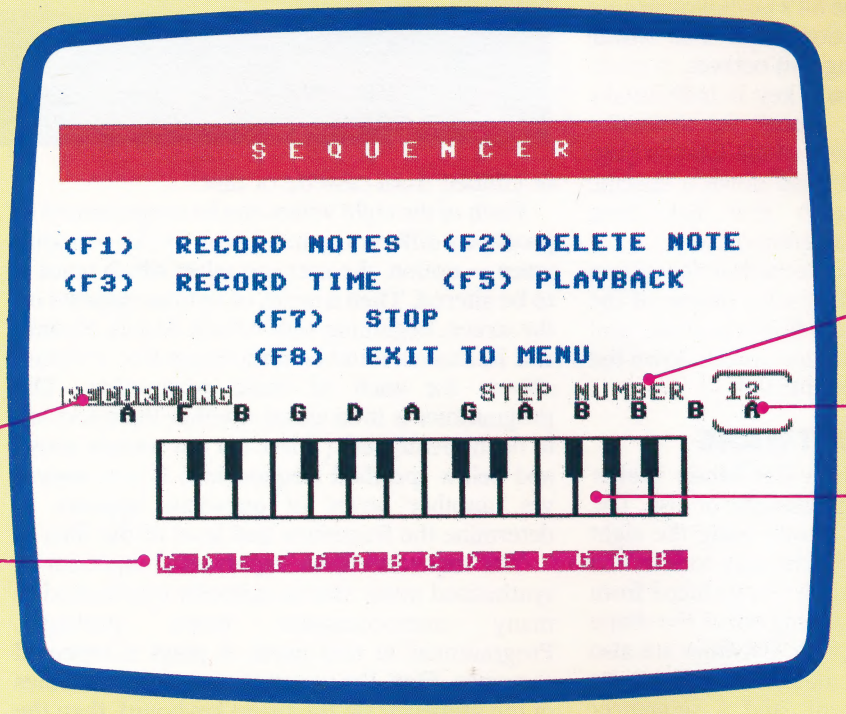
Another difficulty that arises when playing in 'real time' is that parameters such as voice and octave cannot be changed while the user is in 'play mode'. This means that the player is effectively limited to two octaves.

Two manuals are provided with the Music Maker. The user guide is a pamphlet with loading instructions and a brief explanation on how each of the various functions is used. It is not detailed but is enough to get the beginner started. From then on the menu-driven instructions are sufficient to allow the user to develop the full capabilities of the package. The other booklet, *Start Playing Keyboard*, contains brief instructions on how to play keyboards and an explanation of musical notation. The rest of the book contains the scores of 28 popular songs.

The Commodore Music Maker is certainly a worthy attempt to make full use of the sound facilities of the Commodore 64. For someone new to computerised music the package is very simple to use, and once you get used to the keyboard it is easy to play tunes. More advanced users may find that the package lacks versatility and that the keyboard is a little too cramped to allow complex effects. However, the Music Maker is a worthwhile investment for anyone wanting to investigate the possibilities of computer music.

Musical Menu

This screen shows the menu display of the Music Maker's sequencer option.



The menu shows the six available options. Notice that the notes to be played and their duration (TIME) are entered separately

This reminds the user that the Music Maker is in recording mode rather than playback

This line reminds the user of the names of each of the notes on the keyboard

The step number indicates how many notes have been input so far into the sequencer memory

These are the notes that have been entered so far

A blue cursor on the keyboard indicates which note is being played at the time

PUBLIC ADDRESS SYSTEM

We've examined the operating principles behind computer communications and the software and hardware available to turn your micro into a terminal. Here, we profile the UK's three main commercial public access systems — Telecom Gold, Prestel and Compunet — and briefly consider BT's Packet Switching Service.

Public access systems divide into two types: free public access systems, where no charges are levied on the public for using them, and commercial systems. The three most important commercial systems in the UK are Telecom Gold, Prestel (incorporating Micronet 800) and Compunet. We looked at Prestel on pages 101 to 103, so we'll just provide a brief update here. Free public access systems are more commonly known as bulletin boards. These are run on ordinary micros by hobbyists for the benefit of other hobbyists. We focus on the commercial systems here, and will examine free public access systems in detail in the next instalment.

Telecom Gold is British Telecom's own electronic mail system. Aimed primarily at business users, it offers instant electronic mail between Gold users, access to telex facilities and file storage. It costs £100 to buy a 'mailbox' and after the first month (which is free) the subscriber is charged both for time on the system and for units of storage. The minimum charge is £10 a month. The hourly rate is high during office hours but about a third as expensive at other times. Storage charges are made for unread mail and both mail and workfiles saved on disk.

Because Gold is designed with large companies in mind, user 'passwords' comprise a three-letter code identifying the company (known as the 'family') and a three-digit code identifying the individual within the company. Thus, for example, TCG035 refers to Lucy Storer of Telecom Gold, who operates one of the 'help' mailboxes.

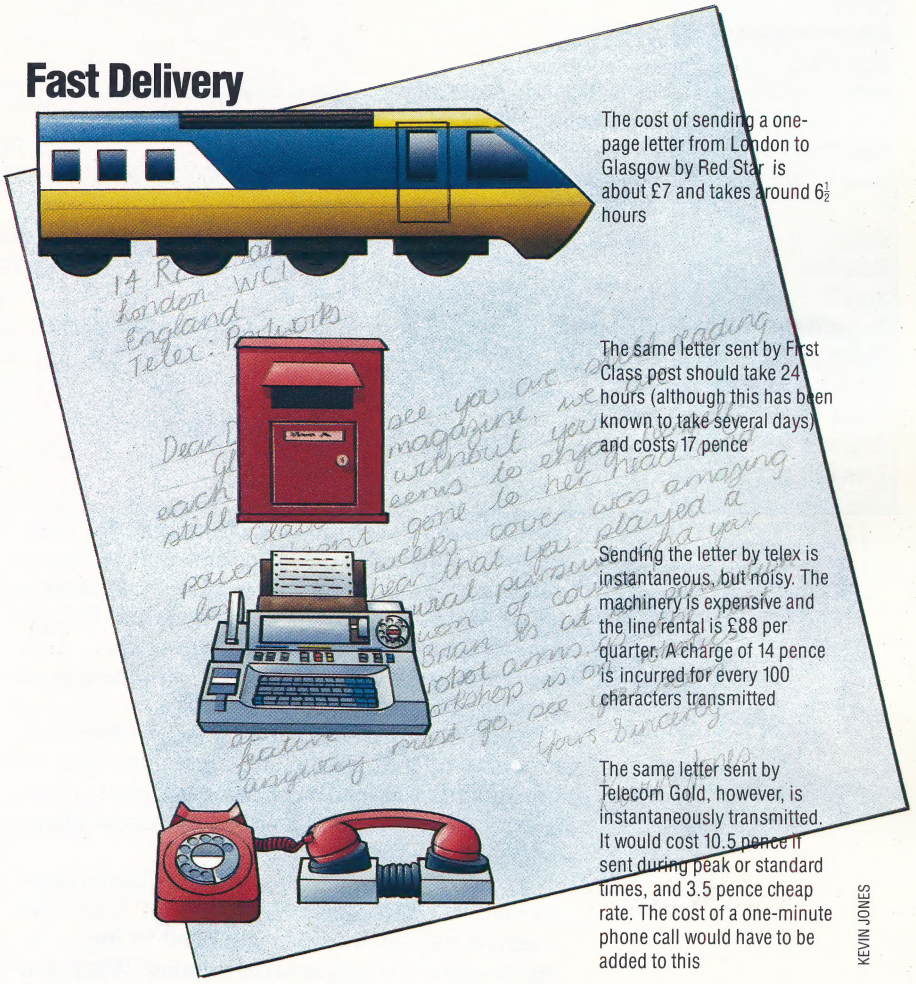
Several computer companies have taken three-letter, three-digit code system mailboxes in bulk at cheap rates and have then sold individual mailboxes to their customers at a reduced rate. Thus, Tandy customers can buy a TCC (Tandy Computer Corporation) mailbox for £20, instead of the full £100. You don't get a month's free use if you buy this way, but unless you're likely to run up charges of more than £80 per month it's worth checking whether your micro's manufacturer offers a similar deal.

Prestel is also run by British Telecom and, like Gold, offers electronic mail facilities, although it is designed primarily to provide information. Prestel is a giant database of information on subjects as diverse as share prices, cookery, travel, top ten software hits, computer clubs, company activities and local weather.

The service operates on a tree structure: you start at the 'root' menu and select a general area — computing, say. You then gradually narrow down your selection through a series of sub-menus until you find what you're looking for — for example, BBC Micro, software, free software, games, Pacman. If you know the 'page' number of the piece of information you want, though, you can skip all the menus and go straight to it.

The main area of interest to computer hobbyist users is known as Micronet 800. This allows you to play on-line games, download software (both free and commercial), leave technical questions and receive answers, and read micro news and reviews. You can also send private electronic mail, although this facility is relatively unsophisticated.

To join Micronet you need a suitable modem and communications software, and for users of the



HCAC Communications Datasheet

System	Telecom Gold	Prestel	Compunet	PSS
Telephone Number (Data)	01-278 4355, or via PSS	Various (depends on area) 300/300-baud = 01-248 5747	Various (depends on area)	Various (depends on area)
Telephone Number (Enquiries)	01-403 6777	01-278 3143	0536-205252	100 (ask for Freefone PSS)
Cost	£100, plus £10 per month (minimum)	About £100 plus £13 per quarter (minimum)	£100 plus £30 per annum	£25 plus £25 per quarter (minimum)
Operating Times	24 hour	24 hour	24 hour	24 hour
Baud Rate(s)	300/300 and 1200/1200	1200/75 (& 300/300 in London)	1200/75	300/300 and 1200/1200
Parity	None	Viewdata	Viewdata	None
Data Bits	8	Viewdata	Viewdata	8
Stop Bits	1	Viewdata	Viewdata	1
To Get Help	INFO INFO	*0£ (*36£ to leave message)	Select HELP from the menu	None available
To Log Off	OFF	Hang up	Select LEAVE from the menu	Hang up
Electronic Mail?	Yes	Yes	Yes	No
Public Messaging?	Yes	Yes	Yes	No
Commercial Software?	No	Yes	Yes	No
Free Software?	No	Yes	Yes	No

BBC Micro, Spectrum or Commodore 64 these may be obtained from Micronet 800 at prices ranging from £75 to £130. Quarterly subscriptions are £13 for home users and £25.50 for company or educational subscribers; then there are time charges to be paid, plus the cost of anything bought through the service.

Compunet provides a similar range of services to those offered in the Micronet 800 area of Prestel. You can send electronic mail, place public messages, download software and access micro news and information.

To access Compunet you need a Commodore modem. This contains a ROM that holds the communications software you need to log on to the system and a unique serial number. When you log on you will be asked for your identification, which is checked against the serial number in your modem's ROM: if the two don't tally Compunet won't allow you access.

This system has both advantages and disadvantages. The obvious advantage to both Compunet and the user is that it doesn't matter if someone else manages to 'hack' your identification code, because they won't be able to use it without your modem. The disadvantage is that you can't log on to the system through someone else's modem; this is inconvenient if your own modem is out of action.

Another protection device built into the system is intended to defeat software pirates. When you download software from Compunet it is 'stamped' with the serial number of the modem used to download it. The software can then be run only with the modem plugged in.

Like Prestel, Compunet is based on a tree structure, with main menus leading to sub-menus. Compunet is 'friendlier' than Prestel, however, as it offers menus in the main area of the screen and a scrolling command menu at the lower part. The command menu gives options such as 'VIEW highlighted option', 'Go back to the LAST page', 'COPY the current page to tape or disk' and so on. The up/down cursor keys are used to select from the Compunet menu, and the left/right keys to select from the command menu.

The Commodore Communications Modem costs £100 (including software) and the Compunet subscription charge is £30 a year, with the first 12 months free. Access costs £7 per hour standard time, but it is free at all other times (evenings, night-time and weekends).

Compunet is not — contrary to popular belief and the impression given by Commodore — Commodore's own viewdata service. It is run by Compunet Teleservices Ltd, an independent company. At present the system supports only the Commodore 64, but Compunet intends eventually to support all the popular home micros.

PACKET SWITCHING SERVICE

Finally, let's take a brief look at British Telecom's Packet Switching Service (PSS), which is not a computer service in the sense that Gold, Prestel or Compunet are, but is a cheap way of accessing other computer systems.

Unless you are within the local calling area of the computer service you are using, the cost of the telephone call can be the main expense. This is especially true if you use American systems, but can also apply to long distance inland calls. PSS is British Telecom's answer to the problem.

PSS is a network connected to all the main public access computer systems worldwide. Users can dial a local PSS 'node' (entry point) and through it access any of the systems connected. The user pays only the cost of the call to the local node (normally a local call) plus a PSS charge. The 'Sample Session Using Telecom Gold' box shows how Gold is accessed via PSS.

When you get the carrier tone, press <CR> <CR> a2 <CR>. This tells PSS that you are a standard CRT, ASCII terminal. PSS will then respond with a piece of code, which will look something like this:

SLO\A01-7531740139

ntlgold <+non-printing password>

You won't be asked to log on; in fact you won't even be given a prompt character! PSS will simply wait for you to enter your Network User Identification (NUI). This is 'n' followed by a 6-character ID and a 6-character password. You enter the lot as one line, but the password won't appear on your screen:

ADD?

If PSS doesn't recognise your ID and password, it will respond with 'NUI?' and wait for you to try again. When your'e on, it will ask you the network address of the service you want, using the prompt:

You then type in the address of the service you want. This will be an 'a' followed by an 11-digit code number. In this case, we want Telecom Gold, which has the code number:

a21920100481

23421920100481+COM

PSS attempts to connect us to this service. If all is well it displays something like:

Primecom Network 18.4G System 81

And you're into the system you've called. We've called Telecom Gold, so let's see how Gold works:

Please Sign On
>id tcc007

To sign on, we have to enter the letters 'id' followed by a six-character identity. This ID will have three letters describing the 'family' of mailboxes (see main text) and three numbers identifying the individual mailbox:

Gold will then prompt for the password. This is 1 to 8 case dependent alphanumeric characters, and is not displayed:

Password: <non-printing password>

Gold then logs you in:

TELECOM GOLD Automated Office Services 18.4G(81)
On At 18:23 30/09/84 BST
Last On At 16:53 27/09/84 BST

Mail call (1 Unread)

Gold tells you if you have any mail in your mailbox and if you have read it:

>mail
Send, Read or Scan: read

To read your mail enter:

Gold tells you who the letter is from, its subject, and the date and time it was posted. It then asks you if you want to read it by presenting a 'MORE?' prompt. Press <CR> to continue, 'n' to skip;

To: TCC007 (81:TCC007)
From: L.STORER (BTG035) Posted: Fri 28-Sep-84
16:46 BST UK Sys 80
Subject: Reply to: query re: 81 to 84 changeover

--More-- <CR>

YES, WHEN I HAVE ALL THE DETAILS I WILL CERTAINLY REPLY

LUCY

You will be asked what you want to do with the letter. You can SAVE it in your personal files, REPLY to it, DELETE it or just leave it where it is. Due to a quirk in Gold, you get charged a storage fee for unread letters. If you want free storage on Gold, simply write yourself a letter and then read it!

Action Required: reply
Text:

I see. The Helpline tells me that Gold will be doing a mail out to all users informing them of the change. Can you tell me when this will be happening? Will everyone on all systems be notified? If so, it shouldn't be necessary for me to have to tell them personally.

Surya

Now let's send the letter:

.send
BTG035 -- Sent

Gold now asks what else we'd like to do with the letter. Since we've finished with it lets delete it:

Action Required: delete
End of Mail

Pressing <CR> drops you out of most unwanted Gold prompts:

Send, Read or Scan: <CR>

>mailck
No mail at this time

If you've been on the system a while, it's a good idea to see if any mail came in while you were on:

We're finished, so let's log off:

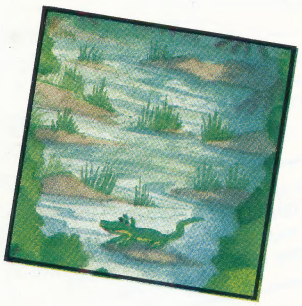
> off
Off At 18:29 30/09/84 BST
Connect Mins = 7
Compute Secs = 4/3

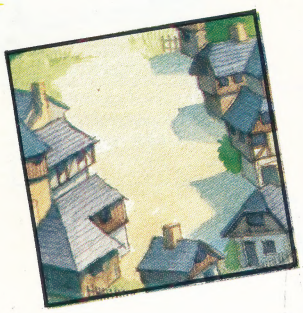
CLR DTE (00) 00:00:06:07 47 49

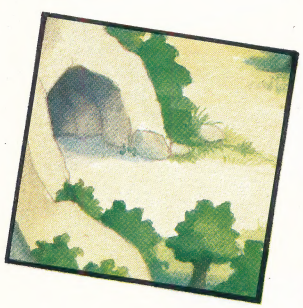
Now we're back in PSS, but you can simply hang up on this:

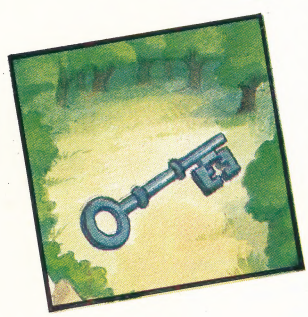
IAN McKINNELL

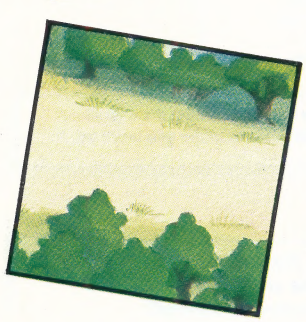
SINK OR SWIM











As our adventure game programming project approaches its conclusion, we analyse the design of the last two special locations in Haunted Forest — the swamp and the village — and complete the listing for our Digitaya adventure.

Two special locations in Haunted Forest are still to be dealt with: the swamp and the village. Let us start by looking at the swamp location. As with all special locations it is important to decide on a storyline for the special location before starting to program. This story can be as complex or as simple as the programmer sees fit, but he must bear in mind that a very complex plot at each location can make for a great deal of programming effort and eat large holes in the computer's residual memory.

THE SWAMP

The storyline for the swamp is as follows:

As the player enters the swamp he starts to sink.
The player can use all the 'normal commands' available, but cannot leave the swamp.
Instead, the player must elect to swim if he is to leave the swamp.
The player can swim only if he carries less than two objects, as these weigh him down.
If the player is carrying two objects, then he must drop one if he is not to sink.
All objects dropped in the swamp are lost for good and cannot be recovered later.

```
4870 REM **** SWAMP S/R ****
4875 SF=1
4880 SN$="YOU START TO SINK INTO THE SWAMP.":GOSUB
5500
4885 PRINT:INPUT"INSTRUCTIONS";IS$
4890 GOSUB2500:REM SPLIT INSTRUCTION
4895 IF F=0 THEN 4885:REM INVALID
4900 GOSUB3000:REM NORMAL INSTRUCTIONS
4910 IF VB$="LOOK" THENGOSUB2000:GOTO4885
4915 IF VB$="DROP"THEN IV$(F,2)="-2":REM OBJ LOST
FOREVER
4917 IF VF=1 THEN 4885:REM NORMAL COMMAND
4920 REM ** NEW COMMANDS **
4925 IF VB$<>"SWIM"THEN SN$="I DON'T UNDERSTAND":G
OSUB5500:GOTO4885
4930 REM ** SWIM **
4932 F=0
4935 FOR I=1 TO 2
4940 IF IC$(I)<>""THEN F=F+1
4950 NEXT I
4955 IF F<2 THENGOSUB5035:RETURN:REM SWIM AWAY
4960 GOSUB 5000:RETURN:REM TWO OBJS HELD
5000 REM **** TWO OBJECTS HELD S/R ****
5010 SN$="THE OBJECTS ARE WEIGHING YOU DOWN AND YO
U ARE SINKING.":GOSUB5500
5012 PRINT:INPUT"INSTRUCTIONS";IS$
5015 GOSUB2500:REM SPLIT INSTRUCTION
5020 IF VB$<>"DROP" THENGOSUB5080:REM SINK
5025 GOSUB3900:IV$(F,2)="-2":REM DROP OBJ
5030 IF HF=0 OR F=0 THEN 5080:REM SINK
5035 REM **** SWIM AWAY ****
5040 SN$="YOU CAN NOW SWIM THROUGH THE SWAMP. WHIC
H WAY WILL YOU GO?":GOSUB5500
5050 EX$(2)="00080605":GOSUB2300:REM DEFINE AND DI
SPLAY EXITS
5055 PRINT:INPUT"INSTRUCTIONS";IS$
5060 GOSUB2500:REM SPLIT INSTRUCTION
5062 IF F=0 THEN 5055:REM INVALID
5065 GOSUB3500:REM MOVE
5067 EX$(2)="00000000":REM ZERO EXIT DATA
5070 RETURN
5075 :
5080 REM **** SINK S/R ****
5085 SN$="YOU SINK INTO THE SWAMP AND DROWN":GOSUB
5500
5090 END
```

Lines 4870-4917 allow normal commands to be dealt with, using the standard subroutines previously designed. It the player elects to drop an object immediately on entering the swamp this will be dealt with by the DROP routine. However, this routine reinstates the dropped object's position in the main inventory IV$(,), using the current value of the location counter, P. This in turn means that, as far as the program is concerned, the dropped object now resides at the swamp location. If we want to lose all trace of an object dropped in the swamp, then, in the case of a player wanting to drop an object here, we must amend the relevant entry in IV$(,).

Remember that IV$(F,2) holds the position of object F. Normally this is either the location number, or -1 if the object is carried by the player. To make the object appear to vanish from the game entirely we need to make IV$(F,2) incapable of being interpreted as a location or of indicating that the player holds the object. In line 4915, IV$(F,2) takes the value -2, F having already had the relevant object designated to it by the DROP routine.

If the player elects to SWIM, the program is allowed to fall through to line 4930. Here, the player's personal inventory is checked to determine how many objects are being carried. If the count is less than two then the program calls the SWIM AWAY subroutine, which allows the player to leave the swamp. If the player is carrying two objects he is given the opportunity to drop one; if he fails to take this course of action he sinks.

The SWIM AWAY routine allows the escaping player to specify the direction in which he wishes to swim. The exit data given for the swamp in the original DATA statements is 00000000, indicating that there are no exits from the swamp. Line 5050 redefines the exit data and calls the subroutine that describes the exits. The player is then allowed to select one exit and thus leave the swamp. Note that the exit data for the swamp is zeroed at the end of this routine, so that if the player re-enters the swamp later there won't be any exits.

THE VILLAGE

The village provides the escape route from the haunted forest, although the player still has to

carry out several tasks before he is allowed to leave the adventure. The storyline for the village location is:

The village is surrounded by a high, apparently unscalable wall.
The gate through the wall is guarded.
To get into the village the player must first kill the guard, using the gun.
The player then has to unlock the gate with the key to escape from the forest.

The village routine actually consists of two parts: first, the guard peril has to be negotiated and, second, the gate has to be unlocked. It is not difficult to envisage a scenario where the player, carrying the gun, arrives at the village and kills the guard, only to find that a key is needed to open the gate. If the player does not have the key, then he will need to leave the village location in search of it. If on the player's return the same description of the village location is given — i.e. that a guard is present — this will not be consistent. A particular feature of the game should be encountered and dealt with only once. If the guard is killed, then that 'characteristic' of the village must be removed.

This is not as difficult as it sounds. A flag can be used to denote whether or not the guard is alive or dead. The variable GF is used for this purpose — its initial value is zero indicating a live guard. If the player kills the guard, then GF is set to 1. The value of GF can be tested on entering the village location to determine whether the guard peril still exists or not. Having killed the guard, the player moves forward to the gate. Instructions can be split and dealt with using the standard subroutines that we have already. If the player has the key and uses it to unlock the gate then he has successfully completed the adventure.

```
5100 REM **** VILLAGE S/R ****
5102 SF=1
5105 SN$="THE VILLAGE IS SURROUNDED BY A TALL WALL
.":GOSUB5500
5106 IF GF<>0 THEN GOSUB5190:RETURN:REM GATE
5107 SN$="A GUARD IS AT THE ENTRANCE GATE TO THE V
ILLAGE":GOSUB5500
5115 PRINT:INPUT"INSTRUCTIONS";IS$
5120 GOSUB2500:IF F=0 THEN 5115:REM INVALID
5125 GOSUB3000:REM NORMAL INSTRUCTIONS
5130 IF VB$="LOOK"THEN GOSUB2000:REM DESCRIBE
5135 IF VB$="GO" AND MF=1 THEN RETURN
5140 IF VF=1 THEN 5115:REM NEXT INSTRUCTION
5145 IF VB$<>"KILL"THENSN$="I DON'T UNDERSTAND":GO
SUB5500:GOTO5115
5150 REM ** KILL **
5155 SN$="WHAT WILL YOU USE TO KILL THE GUARD?":GO
SUB5500
5160 SN$="ENTER OBJECT OR <I> FOR INSTRUCTION":GOS
UB5500
5162 INPUT IS$:IF IS$="I" THEN 5115
5165 GOSUB2500:REM SPLIT
5167 IF F=0 THEN 5160:REM INVALID
5170 GOSUB5300:IF F=0 THEN SN$="THERE IS NO "+W$:G
OSUB5500:GOTO5160
5172 OV=F:GOSUB5450:REM IS OBJECT HELD
5174 IF HF=0 THEN SN$="YOU DO NOT HAVE THE "+IV$(F
,1):GOSUB5500:GOTO5160
5175 IF F<>1 THEN SN$="THE "+IV$(F,1)+" IS NO USE"
:GOSUB5500:GOTO5160
5180 SN$="YOU KILL THE GUARD":GOSUB5500:GF=1
5185 :
5190 REM **** LOCKED GATE S/R ****
5195 SN$="YOU MOVE FORWARD AND TRY TO OPEN THE GAT
E TO THE VILLAGE"
5200 SN$=SN$+" BUT THE GATE IS LOCKED AND WILL NOT
 MOVE":GOSUB5500
5205 PRINT:INPUT"INSTRUCTIONS";IS$
5210 GOSUB2500:IF F=0 THEN 5205:REM INVALID
5215 GOSUB3000:REM NORMAL INSTRUCTIONS
5220 IF VB$="LOOK"THEN GOSUB2000:REM DESCRIBE
5225 IF VB$="GO" AND MF=1 THEN RETURN
5230 IF VF=1 THEN 5205:REM NEXT INSTRUCTION
5232 IF VB$="USE"THEN 5240
5234 IF VB$="UNLOCK"THENSN$="HOW?":GOSUB5500:GOTO5
205
5235 SN$="I DON'T UNDERSTAND":GOSUB5500:GOTO5205
5240 GOSUB5300:REM VALID OBJECT
5242 OV=F:GOSUB5450:REM IS OBJ CARRIED
5244 IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:G
OTO5205
5246 IF HF=0 THEN SN$="YOU DO NOT HAVE THE "IV$(F,
1):GOSUB5500:GOTO5205
5248 IF F<>3 THEN SN$="THE "+IV$(F,1)+" IS NO USE"
:GOSUB5500:GOTO5205
5250 REM ** THROUGH GATE AND SAFE **
5255 SN$="YOU UNLOCK THE GATE, AND DISGUISING YOUR
SELF IN THE DEAD"
5260 SN$=SN$+" GUARD'S CLOTHES, WALK UNNOTICED THR
OUGH THE VILLAGE"
5265 SN$=SN$+" AND THE SAFETY OF THE OUTSIDE WORLD
.":GOSUB5500
5270 END
```

So that we can call these two special location routines, we must amend line 2720 of the subroutine that decides whether or not a location is special. Make this change:

```
2720 ON P GOSUB4590,4870,5100,4590
```

Basic Flavours

Spectrum:
In the Haunted Forest listing, replace SN$ by S$, IS$ by T$, VB$ by B$, IV$(,) by V$(,), EX$() by X$() and IC$() by I$().
Substitute the following lines:

```
2720 IF P=1 THEN GOSUB 4590
2722 IF P=2 THEN GOSUB 4870
2724 IF P=3 THEN GOSUB 5100
2726 IF P=4 THEN GOSUB 4590

4937 LET A$=IC$(I):GOSUB7000
4940 IF A$<>"" THEN LET F=F+1

5175 IF F<>1 THEN LET S$="THE "
     :LET A$=V$(F,1):GOSUB7000
5177 IF F<>1THEN LET S$=S$+"IS NO
     USE":GOSUB5500:GOTO5160

5248 IF F<>3 THEN LET S$="THE"
     :LET A$=V$(F,1):GOSUB7000
5249 IF F<>3 THEN LET S$=S$+
     "IS NO USE":GOSUB5500:GOTO5205
```

In the Digitaya listing, replace SN$ by S$, IS$ by I$, VB$ by B$, IV$(,) by V$(,) and IC$() by I$().
Substitute the following lines:

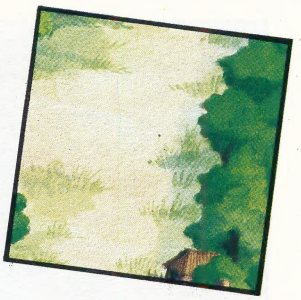
```
3650 LET S$="YOUR ":LET A$=V$(F,1):
     GOSUB 7000
3655 LET S$=S$+" IS USELESS, THE
     FORCE INCREASES"

4520 LET V$(4,2)=STR$(INT(RND(1)
     *40+8)
```

BBC Micro:
In the Haunted Forest listing substitute this line:

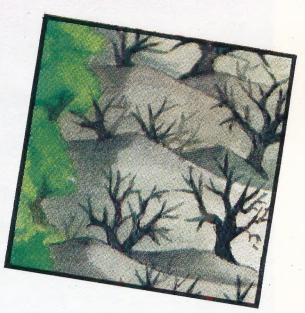
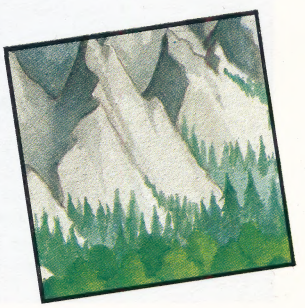
```
190 LET GF=0
```

In the Digitaya listing substitute the following line:

```
4520 IV$(4,2)=RND(40)+8
```





Digitaya Listing

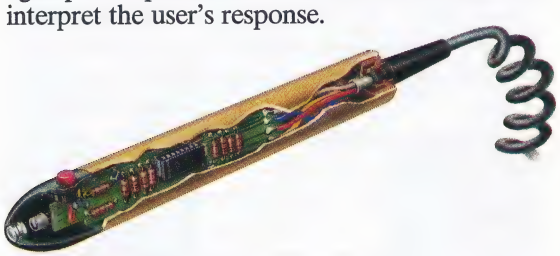
```
2960 REM **** USER PORT S/R ****
2970 SF=1
2980 SN$="ESCAPE IS AT HAND BUT THE DDR BOOKING CL
ERK"
2990 SN$=SN$+" BARS YOUR WAY. HE TELLS YOU THAT HE
 HAS BEEN INSTRUCTED TO"
3000 SN$=SN$+" ACCEPT INPUTS ONLY. HOWEVER HE DOES
 TAKE ALL MAJOR"
3010 SN$=SN$+" CREDIT CARDS."
3020 GOSUB 5880:REM FORMAT PRINT
3030 :
3040 PRINT:INPUT"INSTRUCTIONS";IS$
3050 GOSUB1700:REM ANALYSE INSTRUCTIONS
3060 GOSUB1900:REM NORMAL ACTIONS
3070 IF MF=1 THEN RETURN:REM MOVE OUT
3080 IF VF=1 THEN3040:REM NEXT INSTRUCTION
3090 IF VB$<>"GIVE" THENPRINT"I DON'T UNDERSTAND":
GOTO3040
3100 :
3110 REM ** INSTRUCTION IS GIVE **
3120 GOSUB5730:REM IS OBJECT VALID
3130 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO3040:REM
 NEXT INSTRUCTION
3140 :
3150 REM ** IS OBJECT CREDIT CARD **
3160 IF F<>5THENPRINT"HE ONLY ACCEPTS CREDIT CARDS
":GOTO3040
3170 :
3180 REM ** IS CARD CARRIED **
3190 OV=5:GOSUB5830
3200 IFHF=0THENPRINT"YOU DO NOT HAVE THE ";IV$(5,1
):GOTO 3040
3210 :
3220 SN$="THE CLERK TAKES THE CARD AND SAYS 'THAT
WILL DO NICELY, SIR'"
3230 GOSUB5880:REM FORMAT PRINT
3240 SN$="YOU ARE ALLOWED TO PASS THE BARRIER AND
ENTER THE USER PORT"
3250 GOSUB5880:REM FORMAT PRINT
3260 :
3270 REM ** IS DIGITAYA CARRIED **
3280 OV=6:GOSUB5830
3290 IF HF=1 THEN 3380:REM SUCCESS
3300 :
3310 REM ** FAILURE **
3320 SN$="WELL DONE YOU HAVE SUCCEEDED IN ESCAPING
 FROM THE CLUTCHES"
3330 SN$=SN$+" OF THE MACHINE, BUT HAVE FAILED IN
YOUR MISSION"
3340 SN$=SN$+" TO BRING BACK THE MYSTERIOUS DIGITA
YA"
3350 GOSUB5880:REM FORMAT PRINT
3360 END
3370 :
3380 REM ** SUCCESS **
3390 SN$="CONGRATULATIONS, YOU HAVE SUCCEEDED IN Y
OUR MISSION"
3400 SN$=SN$+" TO RESCUE THE WONDEROUS DIGITAYA FR
OM THE"
3410 SN$=SN$+" CLUTCHES OF THE MACHINE.
3420 GOSUB5880:REM FORMAT PRINT
3430 END
3440 :
3450 REM **** CASSETTE PORT S/R ****
3460 SF=1
3470 SN$="YOU FEEL AN IRRESTISTIBLE FORCE PULLING
YOU TOWARDS"
3480 SN$=SN$+" PERMANANENT MAGNETIC SUSPENSION"
3490 GOSUB5880:REM FORMAT
3500 NS=0:REM START COUNTING INSTRUCTIONS
3510 REM ** INSTRUCTIONS **
3520 NS=NS+1:IFNS>3THEN3770:REM SUCKED OUT
3530 PRINT:INPUT"INSTRUCTIONS";IS$
3540 GOSUB1700:REM ANALYSE INSTRUCTIONS
3550 GOSUB1900:REM NORMAL ACTIONS
3560 IFMF=1THENMF=0:PRINT"YOU CAN'T MOVE...YET":GO
TO3510
3570 IFVF=1THEN3510:REM NEXT INSTRUCTION
3580 IFVB$<>"USE"THENPRINT"I DON'T UNDERSTAND":GOT
O3510
3590 REM ** INSTRUCTION IS USE **
3600 GOSUB5730:REM IS OBJECT VALID
3610 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO3510
3620 :
3630 REM ** IS OBJECT BUFFER ACTIVATOR **
3640 IF F=8 THEN3680:REM OK
3650 SN$="YOUR "+IV$(F,1)+" IS USELESS, THE FORCE
INCREASES"
3660 GOSUB 5880:GOTO3510:REM NEXT INSTRUCTION
3670 :
3680 OV=8:GOSUB5830:REM IS BUFF ACT HELD
3690 IFHF=0THENSN$="YOU DON'T HAVE THE "+IV$(8,1):
GOSUB5880:GOTO3510
3700 :
3710 REM ** SAVED **
3720 SN$="YOU USE THE BUFFER ACTIVATOR TO COUNTER
THE PULL"
3730 SN$=SN$+" INTO MAGNETIC OBLIVION. THE FORCE S
UBSIDES"
3740 GOSUB5880:REM FORMAT
3750 RETURN
3760 :
3770 REM ** SUCKED OUT **
3780 SN$="THE FORCE BECOMES TOO STRONG AND YOU ARE
 PULLED OUT"
3790 SN$=SN$+" THROUGH THE CASSETTE PORT INTO MAGN
ETIC NOTHINGNESS."
3800 GOSUB 5880:REM FORMAT
3810 END
4180 REM **** TRI-STATE DEVICE S/R ****
4190 SF=1
4200 SN$="A LARGE SIGN SAYS 'I/O THIS WAY' BUT AS
YOU MOVE TOWARDS IT"
4210 SN$=SN$+" A TICKET COLLECTOR SHOUTS 'TICKETS
PLEASE'
4220 GOSUB5880:REM FORMAT
4230 :
4240 REM ** INSTRUCTIONS **
4250 PRINT:INPUT"INSTRUCTIONS";IS$
4260 GOSUB1700:GOSUB1900:REM ANALYSE
4270 IFMF=1 THEN RETURN
4280 IFVF=1THEN4240:REM NEXT INSTRUCTION
4290 IFVB$<>"GIVE"ANDVB$<>"OFFER"THENPRINT"I DON'T
 UNDERSTAND":GOTO4240
4300 REM ** INSTRUCTION IS GIVE **
4310 GOSUB5730:REM IS OBJECT VALID
4320 IFF=0THENPRINT"THERE IS NO ";NN$:GOTO4240:REM
 NEXT INSTRUCTION
4330 :
4340 REM ** IS OBJECT TICKET **
4350 IF F=4 THEN4400:REM OK
4360 SN$="THE TICKET COLLECTOR SHAKES HIS HEAD AND
 SAYS"
4370 SN$=SN$+" 'I CANNOT ACCEPT THIS "+IV$(F,1)
4380 GOSUB5880:GOTO4240:REM NEXT INSTRUCTION
4390 :
4400 OV=4:GOSUB5830:REM IS TICKET HELD
4410 IFHF=0THENPRINT"YOU DO NOT HAVE THE TICKET":G
OTO4240
4420 :
4430 REM ** OK **
4440 SN$="THE TICKET COLLECTOR ACCEPTS YOUR TICKET
 AND ALLOWS YOU"
4450 SN$=SN$+" TO PASS THE BARRIER."
4460 GOSUB5880:REM FORMAT
4470 REM ** DEL TICKET FROM LIST **
4480 F=0
4490 FORJ=1TO4
4500 IF IC$(J)=IV$(4,1)THENIC$(J)="":J=4
4510 NEXT J
4520 IV$(4,2)=STR$(INT(RND(TI)*40+8)):REM REALLOCA
TE TICKET POSITION
4530 P=15:MF=1:RETURN
```

LIGHT PEN

This is a device that enables you to interact with a program without using a keyboard. By pressing the *light pen* on a specific position on the monitor or television screen the user can instruct the computer to perform an action, draw a line or select a menu option.

A light pen consists of a tube connected by a cable to a port on the computer. At one end of the tube is a lens, behind which is a photoelectric cell. Light falling on the cell will cause a greater flow of electricity through the circuit. Current from the cell is passed to an amplifier and then transmitted to the computer. Most light pens have an on/off switch and, depending on what kind of port the pen is plugged into, may have an encoding circuit to turn the analogue signal from the cell into a digital one.

The pen works by detecting the presence of the electron beam inside the television or monitor that traverses the screen to build up an image line by line. The video chip inside the computer 'knows' when the beam starts to scan the screen. After a period of time, the chip will receive a signal from the light pen telling it that the beam has passed the pen's photosensitive end. By dividing the speed of the moving beam by the time taken to reach the light pen, the computer can calculate where the light pen is positioned on the screen, and thus interpret the user's response.



Light Lines
A light pen sends information to the computer, which allows it to calculate where the pen is pointing at a VDU or screen. This is useful in applications that involve screen 'drawing' or selecting options from an on-screen menu

LINE PRINTER

A line printer is a device that prints lines of text on paper. Various types of line printer have been developed, including dot matrix, daisy wheel and laser printers. Whatever method is used to put print onto paper, however, these variations have many features in common. Information to be printed arrives from the computer and is stored as data in a buffer (an area of RAM). The buffer stores the text because the computer feeds it into the printer faster than it can be printed. So rather than have the computer waiting for each character to be printed, it delivers information in 'buffer loads', and is thus able to get on with other tasks while the information in the buffer is being printed. When the buffer is empty the printer sends an interrupt signal to the computer, which then delivers another buffer load of data.

A line printer has a typewriter-style carriage around which the paper is fed (usually in a continuous stream) past the print head. The rolling movement of the carriage and the special movements of the print head, such as RETURNs from right to left of the paper-width, are governed by 'control characters'. These are special codes that the printer recognises as commands.

LISP

LISP is an acronym for *LIS*t *P*rocessing. It is a computer language that is based — as its name suggests — on the manipulation of lists of data. The lists are set up within the system and names assigned to them. Elements of a list can then be operated on — for example, a test could be made to see whether a condition is true or false.

Programming in LISP consists of defining the functions with which we can perform our operations. LISP is a 'functional' language, which means that complex functions can be built up from the basic set of LISP commands in much the same way as LOGO. Because of its ability to manipulate words and test them for sense and syntax, LISP is currently the subject of a great deal of work in the field of artificial intelligence.

LOCAL AREA NETWORK

A *local area network* (LAN) consists of a group of computers linked together in such a way as to share a common peripheral device and communicate with each other. Local area networks do not have to consist of the same type of computer, nor do they have to be in the same room or even the same building. Some computers in LANs are in different parts of the world.

There are three types of local area network. A *star* network consists of several computers joined to a single central computer that controls the network and peripherals. Data sent from one satellite computer to another will pass through the central computer.

A *ring* network will have the computers 'daisy-chain' in a continuous loop. Ring networks are unsuitable in large configurations because of the number of machines that the data may have to pass through to reach the target computer.

The third type, *bus* networks, are typified by the Econet system of the BBC Micro. The computers are linked via individual branch cables to a central 'bus', thus allowing any one of them to communicate with any other directly without having to pass through a third computer.

LOGIC

Logic is the fundamental basis of the operation of computers. It is the science by which conclusions are reached by reasoning or inference from a set of parameters. Using such reasoning, logic can be expressed in algebraic form — e.g. IF A THEN B. Logic does not deal in shades of grey but in black and white. A proposition is either true or false. Thus logic, reduced to the essentials of True or False and On or Off, is particularly suited to the binary arithmetic used in computers.

The transistors within a computer can be arranged in such a way as to perform 'logical' operations. There are six of these logical circuits and they can perform AND, NAND, NOT, NOR, OR and Exclusive-OR operations. These circuits are the basis on which the entire computer is built. Thus logic can also refer to any computer circuit.

CHIP CHAT

A Child's First Words
My Talking Computer is intended to be a child's first introduction to computers. It is designed to be as simple to use as possible. Overlays contained in a flip-over book are placed over a touch-sensitive pad and held in place by a frame. Expansion modules containing extra programs can be easily inserted into the slot on the left-hand side of the computer

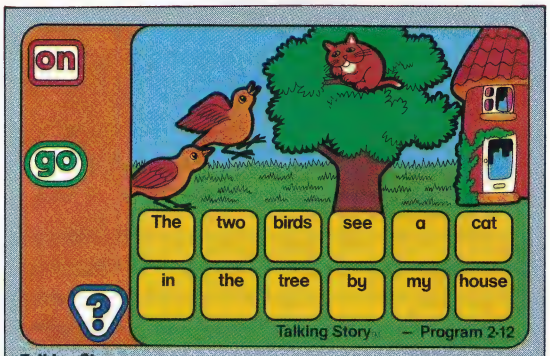
My Talking Computer is a microprocessor-based learning aid for pre-school children, which neatly and economically solves the problem of how infant fingers can communicate via a 'keyboard'. We assess its value as an educational tool — and decide whether it's really a computer at all!

My Talking Computer from Microspeech is intended for use by children of three and upwards, and the company claims that its machine is already in use in primary schools throughout the United Kingdom. The 'computer' is contained in a light plastic case measuring 23 by 25 cm (9 by 10 in). Overlays for the various programs can be placed on a touch-sensitive panel on the front of the machine and these are held in place by a frame. In this respect the machine is similar to the Touchmaster pad (see page 830), although My Talking Computer has much fewer contact points. Already fitted is My Talking Clock which, used in conjunction with one of the built-in programs, teaches children to tell the time.

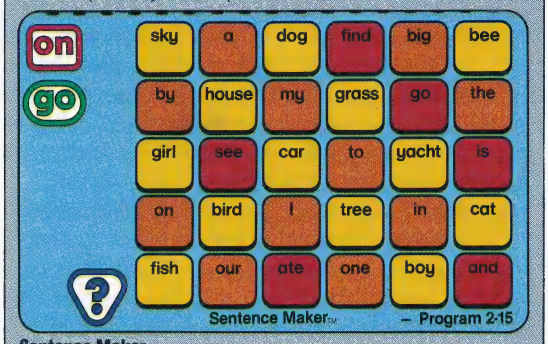
At the side of the machine is a slot where the program modules (cartridge-like devices) are fitted. Although My Talking Computer is an ingenious new approach to teaching small children, the resident programs are obviously limited in their application, which necessarily shortens the useful life of the machine as far as the individual child is concerned. However, the manufacturers attempt to overcome this problem by providing an expansion slot that enables extra ROMs to be fitted. Although the first of these ROMs, Expansion Module One, appears to be pitched at only a slightly higher level than the on-board ROM, the intention seems to be to produce a series that will increase in sophistication as the child gets older. Above the expansion slot is a small speaker. The machine can be run using either five 1.5v batteries or an external power supply.

The programs are held on a single ROM chip that Microspeech claims contains more than 120 Kbytes of data, although this data will not be loaded by the computer all at once. The ROM-based software is divided into five master programs, each of which is subdivided into several programs based around the same subject. The first master program is dedicated to teaching arithmetic, with exercises such as number recognition, and simple addition, subtraction, multiplication and division. The second teaches the relationship between pictures and words. The third is a 'talking calculator' that speaks the figures as you enter them and tells you the result. The fourth program consists of 'talking' games that test a child's ability to respond to relationships, couched in such terms as 'find the dog'. The fifth program, which we have already mentioned, teaches the child how to tell the time.

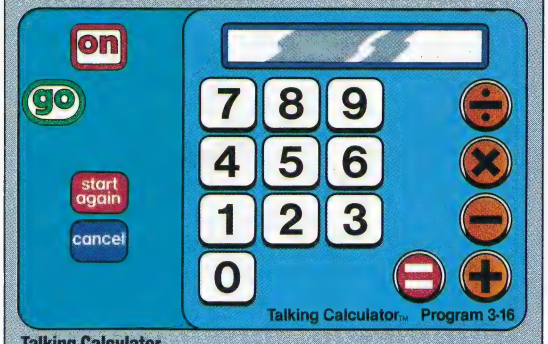
The child uses a particular program by turning through a ring-bound book of overlays and fitting the selected one under the frame on the front of the computer. Pressing the ON square on the overlay switches the machine on and a female voice responds with 'Hello'. The machine then



Talking Story
This overlay is designed for 2 to 12 year olds. The computer recognises which program is being used by the unique positions of the ON and GO 'buttons'. When a word square is pressed, the word is spoken by the computer



Sentence Maker
This program teaches word recognition. When the child presses the question mark, the computer speaks a word, and the child must press the corresponding key



Talking Calculator
One of the more interesting programs available on My Talking Computer is the Talking Calculator. By pressing the numbers and the arithmetic symbols, simple sums can be calculated

IAN McKINNELL

Telling The Time
The computer has My Talking Clock fitted above the touch pad. On the underside of the clock are a set of protruding teeth, which correspond to the pad on the computers, and make the device more accurate. The hands of the clock can be moved around the dial and the computer will tell you what time is being displayed

prompts the child to press the GO key on the overlay. The computer 'knows' which program has been selected because the ON and GO keys are in a different position on each overlay. This method of operation is particularly useful for small children.

Once a child has learnt that ON starts the computer and GO starts the program, he can then load and run any of the programs without any assistance from an adult. The overlays are plastic-coated, which means that jam and chocolate can be wiped off.

When assessing the educational value of a piece of hardware or software, certain criteria apply. The package must teach what it claims to, for example, and it must not assume any prior knowledge of the subject it is trying to teach — pages of written instructions are clearly unacceptable in a 'learn to read' program. It also helps if it is easy and entertaining to use.

THE SPEECH SYNTHESISER

The built-in speech synthesiser is the best thing about the computer. In fact, it makes you wonder why calculators don't have built-in speech facilities. Nevertheless, it has its limitations. The speech is stored as whole words, which are then individually accessed to build the phrases used. The stilted phrasing that results can't really be avoided as the same words are used in a number of contexts. The female voice used has an American accent, which may cause British parents to complain that their children are being taught to speak 'American', but this may annoy adults more than children. The most serious problem, compounded on a machine designed for educational applications, is that many of the words are indistinct: it is difficult, for example, to tell the words 'by', 'my', and 'sky' apart. Even after repetition of certain phrases it is difficult to determine what has been said.

We can now judge whether My Talking Computer meets the criteria we defined as being essential in an educational aid. The ease with which new programs can be fitted and run is certainly to be commended. The fact that the computer actually speaks is a tremendous advantage in teaching a child to read, and neatly avoids the necessity for complicated instruction sheets. Despite doubts concerning the quality of the speech, the computer's ability to associate the sound of the word with its written form is a much more direct method of teaching than that offered by software packages, which simply associate the word with pictures — although, of course, My Talking Computer does that as well. Its ability to respond verbally ensures that it is entertaining to use, as well.

My Talking Computer is obviously not a computer in the usual sense of the word, since it is not 'programmable' by the user. It does, however, show microcomputer technology being applied in educational areas. As well as being a sophisticated toy, it can be used to back up other lessons.

MY TALKING COMPUTER

PRICE
£59.95

DIMENSIONS
250×230×70 mm

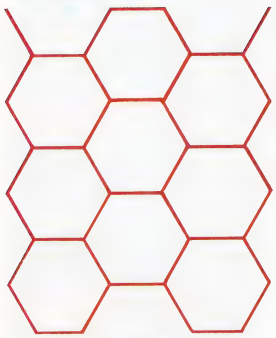
SOFTWARE
Software is provided in ROM on board the computer. Extra software is provided in the form of Expansion Module 1 (£17.95), which plugs into a cartridge slot on the machine

STRENGTHS
My Talking Computer is extremely easy to use. The simple loading procedure and the speech synthesis facility mean that a child does not need to be able to read before using the machine

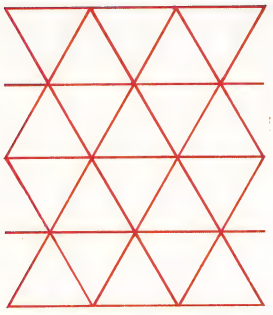
WEAKNESSES
The speech is often indistinct and stilted. The machine is not programmable, and cannot be classified as a true microcomputer

END OF THE LINES

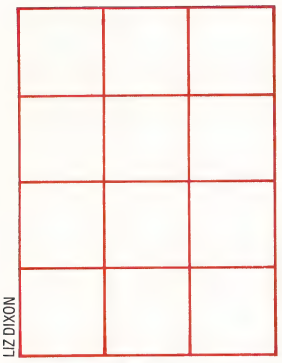
Three Tessellations



Using Hexagonal Tiles



Using Equilateral Triangle Tiles



LIZ DIXON

Using Square Tiles

A 'tessellation' covers an area of a plane by fitting together 'tiles' of the same shape, so that there are no spaces between the individual pieces — an everyday example is the tiling in a bathroom or kitchen. In this final instalment of our series, we show how LOGO is used to tessellate a variety of shapes.

Some regular polygons will form tessellations: squares, equilateral triangles and hexagons (as shown in the margin). However, none of the other polygons tessellate. To see why not, let's consider an example — a pentagon. The angle at the corner of a regular pentagon is 108°. Put three of these together on a focal point and you have used up only 324° — so there is a gap. Add a fourth and the total angle is now 432°, which means that the shapes overlap. Squares, equilateral triangles and hexagons all tessellate precisely because the angles at their vertices (90°, 60° and 120° respectively) divide into 360 an exact number of times.

It is quite straightforward to write LOGO procedures to produce these tessellations. Probably the best approach is to write a procedure to draw the motif starting from the centre and returning to the centre (this involves state transparency — see page 564). We can then use this within a 'superprocedure' that simply moves the turtle to the centre of the next shape to be drawn each time.

The following procedures draw a simple tessellation of squares. Tessellations of equilateral triangles and hexagons could be drawn in a similar way.

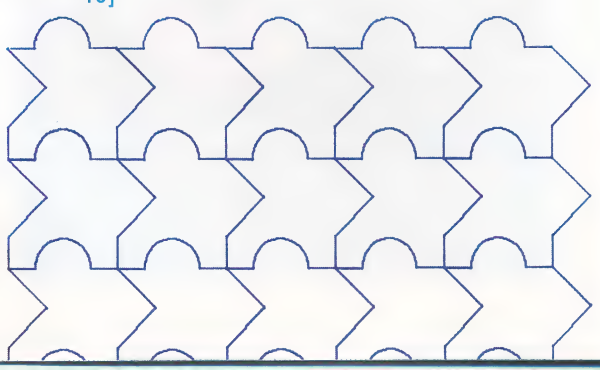
```
TO TESS
   PENUP SETXY (—100) (90) PENDOWN
   REPEAT 5 [VERTLINE SETXY (—100) (YCOR —
   40)]
END

TO VERTLINE
   REPEAT 5 [SQ 40 SETX XCOR + 40]
END

TO SQ :S
   PENUP LEFT 45
   FORWARD :S * (SQRT 2) / 2
   RIGHT 135 PENDOWN
   FORWARD :S RIGHT 90 FORWARD :S RIGHT 90
   FORWARD :S RIGHT 90 FORWARD :S RIGHT 90
   PENUP LEFT 135
   BACK :S * (SQRT 2) / 2 RIGHT 45
END
```

Tessellations can be made from more complex motifs than regular polygons, and can also be made by using a mixture of shapes. Nevertheless, tessellations based around regular polygons have a great deal of potential. Many of M C Escher's drawings are variations on simple regular tessellations.

A straightforward method of building up more complex diagrams involves modifying SQUARE by replacing the FORWARD commands that draw the sides of the square with procedures. The one rule we must observe is that any modification to the top side of the square must be matched by a corresponding modification to the base, and any modification to the right-hand side must be matched by one to the left-hand side, so that the shapes will still fit together.

To draw the basic shape, we simply replace the commands that draw the sides of the square by procedures to draw the new lines. The new tessellation (shown below) is then drawn by calling TESS again. Here is the full listing:

```
TO SQ :S
   PENUP LEFT 45
   FORWARD :S * (SQRT 2) / 2
   RIGHT 135 PENDOWN
   SIDEA :S RIGHT 90 SIDEB :S RIGHT 90
   SIDEC :S RIGHT 90 SIDED :S RIGHT 90
   PENUP LEFT 135
   BACK :S * (SQRT 2) / 2
   RIGHT 45
END

TO SIDEA :S
   FORWARD :S / 4 LEFT 90
   REPEAT 19 [FORWARD 2 * 3.1416 * :S / 144
   RIGHT 10]
   LEFT 100 FORWARD :S / 4
END

TO SIDEB :S
   LEFT 45 FORWARD :S / 2
   RIGHT 90 FORWARD :S / 2
   LEFT 45 FORWARD :S — :S * (SQRT 2) / 2
END

TO SIDEC :S
   FORWARD :S / 4 RIGHT 90
   REPEAT 19 [FORWARD 2 * 3.1416 * :S /144 LEFT
   10]
```



```
    RIGHT 100 FORWARD :S / 4
END

TO SIDED :S
    FORWARD :S — :S * (SQRT 2) / 2 RIGHT 45
    FORWARD :S / 2 LEFT 90
    FORWARD :S /2 RIGHT 45
END
```

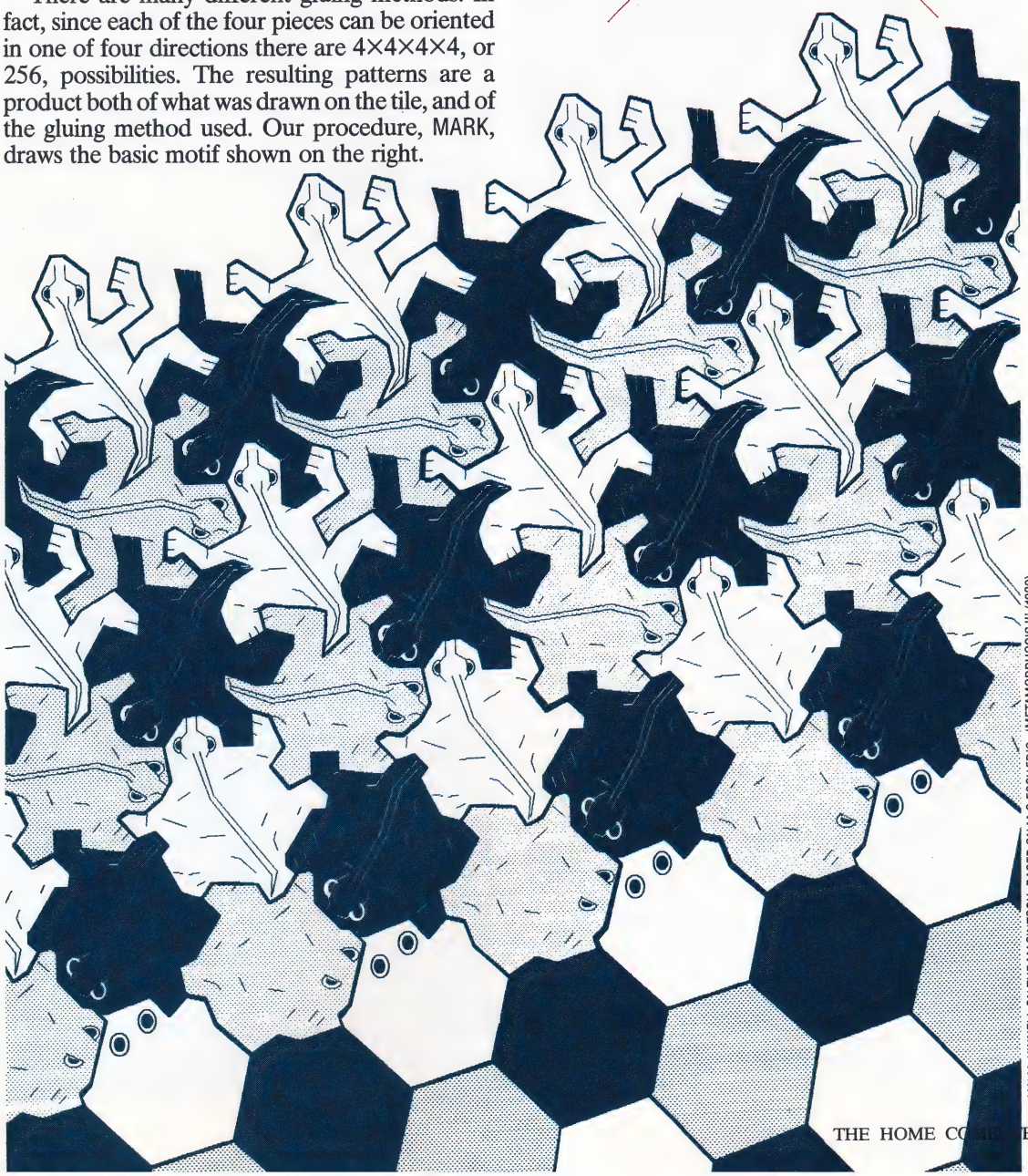
Escher's technique was more complex, of course, as he also slowly modified the shapes as they 'moved' across his tessellation. A LOGO procedure to do that would be very interesting...

USING GLUE

A rather different way to approach the question of tessellations is given by Harold Abelson and Andrea diSessa as an exercise in their excellent book *Turtle Geometry*. Their approach goes like this: think of a shape drawn within a square, say a pattern on a tile, and then glue four of these together to make a larger square. Take this larger square and glue four of these together, and so on.

There are many different gluing methods. In fact, since each of the four pieces can be oriented in one of four directions there are 4×4×4×4, or 256, possibilities. The resulting patterns are a product both of what was drawn on the tile, and of the gluing method used. Our procedure, MARK, draws the basic motif shown on the right.

```
TO MARK :S
    PENDOWN LEFT 45
    FORWARD :S * (SQRT 2) / 2
    BACK :S * (SQRT 2) / 2
    LEFT 90 FORWARD :S * (SQRT 2) / 2 PENUP
    BACK :S * (SQRT 2) PENDOWN
    FORWARD :S * (SQRT 2) / 4 LEFT 45
    FORWARD :S / 2 LEFT 45
    FORWARD :S * (SQRT 2) / 4 PENUP
    BACK :S * (SQRT 2) / 2 LEFT 135
END
```

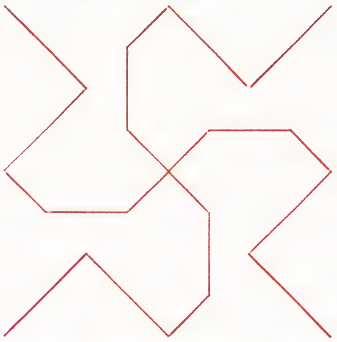


IAN McKINNELL ON THE MACINTOSH: BASED ON M.C.ESCHER'S 'METAMORPHOSIS II' (1939)

Evolving Figures
Tessellating figures are shapes that fit together exactly to cover a plane. 'Metamorphosis II' by M.C. Escher (1898-1972) — on which our computer-created graphic is based — demonstrates how a simple tessellating figure, such as a regular hexagon, can be progressively changed to produce a more complex tessellation — in this case a sophisticated lizard shape. Escher's technique underlines the gradual change from the simple to the complex as the eye moves across the scene

One possible gluing, GLUE1, follows. To run it, type DRAW PENUP GLUE1 [MARK] 100. The resulting pattern is shown.

```
TO GLUE1 :PROC :S
   SQUARE.PIECE 0 :PROC :S
   SQUARE.PIECE 0 :PROC :S
   SQUARE.PIECE 0 :PROC :S
   SQUARE.PIECE 0 :PROC :S
END
```



SQUARE.PIECE draws one of the four squares making up the larger square. It takes three inputs: :A determines the orientation of the square, :PROC is the name of the motif drawing procedure and :S is the length of the side. It is not practicable, given the limits of the screen, to keep on expanding the size of the squares, so the procedure takes the size of the outer square as input and calculates the size of the smaller squares.

```
TO SQUARE.PIECE :A :PROC :S
   FORWARD :S / 4 RIGHT 90
   FORWARD :S / 4 RIGHT 90 * :A
   RUN SENTENCE :PROC :S / 2 LEFT 90 * :A
   BACK :S / 4 LEFT 90
   BACK :S / 4 RIGHT 90
END
```

Here are a number of other possible gluings:

```
TO GLUE2 :PROC :S
   SQUARE.PIECE 0 :PROC :S
   SQUARE.PIECE 1 :PROC :S
   SQUARE.PIECE 2 :PROC :S
   SQUARE.PIECE 3 :PROC :S
END

TO GLUE3 :PROC :S
   SQUARE.PIECE 0 :PROC :S
   SQUARE.PIECE 2 :PROC :S
   SQUARE.PIECE 3 :PROC :S
   SQUARE.PIECE 1 :PROC :S
END

TO GLUE4 :PROC :S
   SQUARE.PIECE 3 :PROC :S
   SQUARE.PIECE 2 :PROC :S
   SQUARE.PIECE 1 :PROC :S
   SQUARE.PIECE 0 :PROC :S
END

TO GLUE5 :PROC :S
   SQUARE.PIECE 2 :PROC :S
   SQUARE.PIECE 1 :PROC :S
   SQUARE.PIECE 0 :PROC :S
   SQUARE.PIECE 3 :PROC :S
END
```

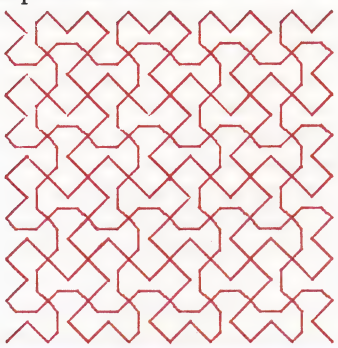
MARK is the name of a shape-drawing command that requires one input. We glued this together with GLUE1 [MARK] 100. Looking at this command line we can see that GLUE1 [MARK] can itself be thought of as a command that only requires a number in order to draw a shape. We can, therefore, use [GLUE1 [MARK]] as the input to another GLUE command. For example:

```
GLUE2 [GLUE1 [MARK]] 100
```

The fun is only just beginning! Consider:

```
GLUE3 [GLUE2 [GLUE1 [MARK]]] 100
```

We now have 256×256×256 combinations of three-level gluings to consider. Nor is there any need to stop at three levels!

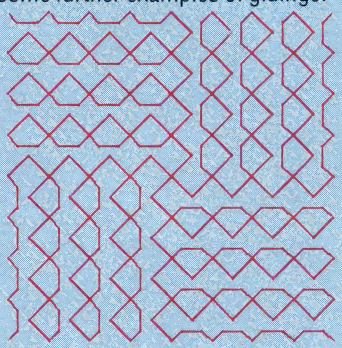


Logo Flavours

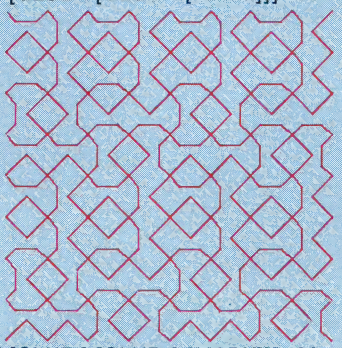
In all LCSI versions, use SETPOS instead of SETXY, and remember it must be followed by a list.
In Atari LOGO you must use the abbreviation SE for SENTENCE

Stuck For Ideas?

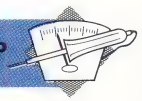
Here are some further examples of gluings:



GLUE 1 [GLUE 4 [GLUE 5 [MARK]]] 100



GLUE 4 [GLUE 3 [GLUE 3 [MARK]]] 100



RECONNAISSANCE MISSION

We continue to investigate a variety of applications for the Workshop robot that we have built. Here, we design a piece of software for the robot to scan a given area and display the shape of an object found within the area.

To allow our robot to scan a given area, we must first devise a scanning pattern that will cover the area efficiently. We may start by considering a scanning pattern where the robot moves backwards and forwards across the area probing for an object. On locating an object, we could elect to probe around all of its sides before proceeding to 'sweep' the rest of the area. However, such an algorithm is difficult to program, and can lead to problems if more than one object is present within the area to be scanned. An alternative course of action on meeting the object is simply to turn around and continue to scan as before. The steps of such an algorithm can be described as follows:

```
REPEAT
    REPEAT
    Probe forwards for an object
    UNTIL side edge of defined area is reached or
        object is found
Move up one 'strip-width'
Turn around
UNTIL top edge of defined area is reached
```

On completion of this horizontal scan, the robot will not have probed into every area available to it if an object is present. Using this scanning pattern, the area behind any object (the 'blind area') will remain unscanned. At least one vertical scan (i.e. at 90° to the first) is required.

SENSOR MODIFICATION

Before the Workshop robot will perform a successful scan, a minor modification must be made to the two front sensors. Because the robot's wheels protrude out the sides, there is a danger that these will come into contact with an object as the robot scans past its side. To get around this problem we must add a shield to the front of each sensor, to guard the wheels. Each shield should be a rectangle, approximately 95 by 25mm ($3\frac{3}{4}$ by 1in), made of a light-weight material, such as rigid plastic or metal. Make sure that the material used is not so heavy that it pulls the sensor down into the closed position. The shields can be mounted onto the sensor tips using tape or glue, and should be positioned so that they extend outwards to cover the wheels exactly. Check the action of the sensor switch to ensure that it can still open and close.

SHAPE SCANNER PROGRAM

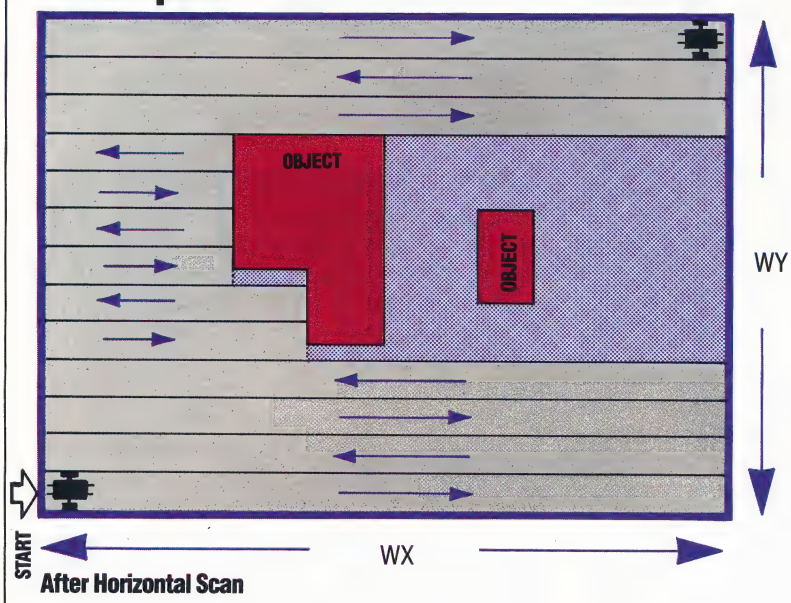
The program used to make the Workshop robot perform an area scan has been written using a single horizontal and a single vertical scan pattern. As the robot completes its traverse of each strip of the scan, a corresponding area of the computer's screen is filled with colour. At the end of the scan a pictorial representation of the area — as the robot 'sees' it — will be displayed on the screen.

Both versions of our program ask for the user to type in the dimensions of the area to be scanned. Using BBC Mode 4, or the Commodore 64's high-resolution display, each pixel on the screen is made to correspond to a 4 mm square in the area being scanned. Maximum dimensions in the horizontal and vertical directions are therefore 1,279 and 1,023 mm respectively, for both machines. Note that a strip — width of 40 mm ($1\frac{1}{2}$in) is used by each program.

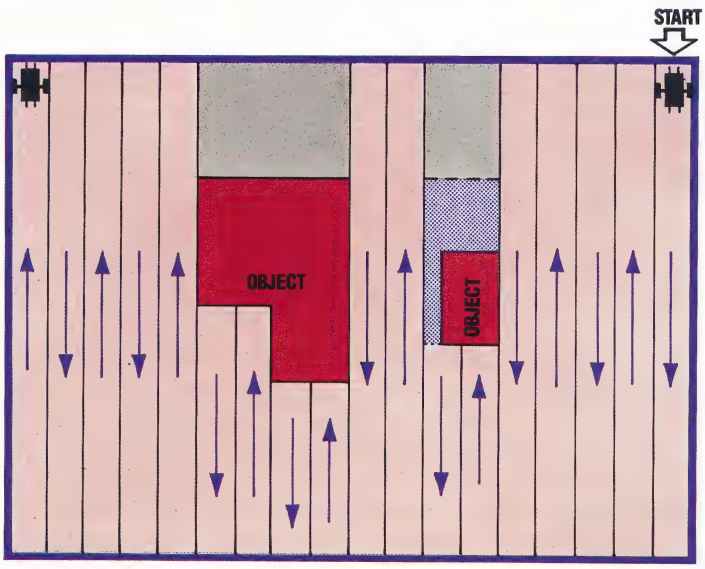
As the robot completes each strip during the scan, the graphics display is updated, using the procedures (or subroutines) called XPLOT and YPLOT. In the BBC Micro's version, the strip area on the screen is simply filled by using the MOVE and DRAW commands to produce a series of adjacent parallel lines. The Commodore 64 version, however, relies on the two machine code routines — PLOTSUB and LINESUB — designed earlier in the course to make up for the lack of high-resolution drawing commands in Commodore BASIC (see pages 337 and 416 respectively). You may have copies of the final object files for these two routines, which can be loaded using lines 30 and 40 of the Commodore version of the program. Alternatively, you may have BASIC loaders for these two routines. Each routine should be separately loaded and run, before loading and running the program. In this latter case, lines 30 and 40 can be omitted.

Apart from the substantial blind area that can be left after two scans, there are several other difficulties that can occur. Most prominent among these is that the program works with varying degrees of success for different shapes or different orientations. It will be most successful with rectangular shapes, orientated along the horizontal and vertical axes. Triangular and circular shapes will produce larger blind areas, as the scan pattern will tend to 'square off' irregularities in the objects that are met. You may wish, as an exercise, to extend the program given here so that a further vertical scan is made. The robot can also get into difficulties if it encounters an object during its manoeuvre to begin a new strip. For simplicity's sake, no check is made on the state of the sensors during this manoeuvre.

Blind Spots



After Horizontal Scan



After Vertical Scan

Key:

Horizontally scanned area

Vertically scanned area

Unscanned area

The simplest method for scanning a given rectangular area is to make two 'sweeps' at 90° to each other. Whenever an object is met, or the edge of the area reached, the robot moves up a strip and continues the sweep in the opposite direction.

Using this method, 'blind areas' will be left behind any objects found. The second sweep will, however, fill in much of the detail that was 'unseen' during the first sweep. Inaccuracies will still arise because of the width of the sweep strip used and the mutual orientation of objects

KEVIN JONES

BBC Micro Listing

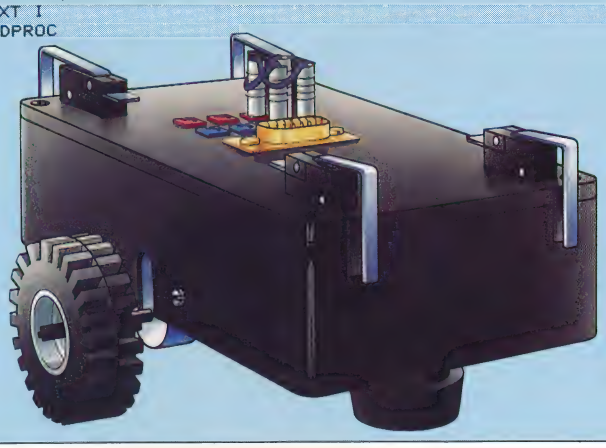
```
 10REM **** BBC SHAPE SCANNER ****
 20MODE 4
 30PROCinitialise
 40PROCscan_dimensions
 50PROChoriz_scan
 60PROCvert_scan
 70END
 80DEF PROChoriz_scan
 90 target=wx:sense=left:xstart=0
100REPEAT
110REPEAT
120PROCmove(forwards,dx):x=x+dx
130UNTIL(?DATREG AND 192)<>neither_bumpers OR x=target
140PROCxplot
150dx=-dx:y=y+dy:xstart=x
160IF target=0 THEN target=wx ELSE target=0
170IF y<wy THEN PROCnext_strip(sense)
180IF sense=right THEN sense=left ELSE sense=right
190UNTIL y>=wy
200ENDPROC
210:
220DEF PROCvert_scan
230IF x=0 THEN sense=left:dx=width ELSE sense=right:dx=-width
```

Commodore 64 Listing

```
10 REM **** CBM SHAPE SCANNER ****
20 DN=8:REM IF CASS THEN DN=1
30 IF A=0 THEN A=1:LOAD"PLOTSUB.HEX",DN,1
40 IF A=1 THEN A=2:LOAD"LINESUB.HEX",DN,1
50 GOSUB1000:REM INITIALISE
60 GOSUB2000:REM SCAN DIMENSIONS
70 GOSUB5000:REM SWITCH TO HIRES MODE
80 GOSUB3000:REM HORIZONTAL SCAN
90 GOSUB4000:REM VERTICAL SCAN
100 GOSUB5100:REM OUT OF HIRES MODE
110 END
120 :
1000 REM **** INITIALISE S/R ****
1010 DDR=56579:DATREG=56577
1020 POKE DDR,15:POKE DATREG,1
1030 FW=4:BW=2:LF=6:RT=0
1040 PD=3.34446:PA=375/90
1050 RB=128:LB=64:BB=0:NB=192
1060 WD=40:DW=WD/10:X=0:Y=0:DX=DW:DY=WD
1070 REM ** M/C START ADRRESSES **
1080 HIRES=49422:LINESUB=49934
1090 RETURN
1100 :
```

BBC Micro Listing Continued

```
240dy=-dw:target=0:ystart=y
250PROCturn(sense,90)
260REPEAT
270REPEAT
280PROCmove(forwards,dy):y=y+dy
290UNTIL(?DATREG AND 192)<>neither_bumpers OR y=target
300PROCyplot
310dy=-dy:x=x+dx:ystart=y
320IF target=0 THEN target=wy ELSE target=0
330PROCnext_strip(sense)
340IF sense=right THEN sense=left ELSE sense=right
350UNTIL x>=wx OR x<0
360ENDPROC
370:
380DEF PROCnext_strip(way)
390PROCmove(backwards,30)
400PROCturn(way,90)
410PROCmove(forwards,width)
420PROCturn(way,90)
430ENDPROC
440:
450DEF PROCscan_dimensions
460INPUT"X DIMENSION IN MM";wx
470 IF wx>1279 THEN 460
480INPUT"Y DIMENSION IN MM";wy
490 IF wy>1023 THEN 480
500wx=width*(wx DIV width)
510wy=width*(wy DIV width)
520CLS
530ENDPROC
540:
550DEF PROCinitialise
560DDR=&FE62:DATREG=&FE60
570?DDR=15:REM LINES 0-3 OUTPUT
580?DATREG=1:REM TURN ON RESET BIT
590forwards=4:backwards=2:left=6:right=0
600pd_ratio=3.34446:pa_ratio=375/90
610right_bumper=128:left_bumper=64
620both_bumpers=0:neither_bumpers=192
630width=40:dw=width/10
640 x=0:y=0:dx=dw:dy=width
650MOVE 0,0
660ENDPROC
670:
680DEF PROCmove(dir,distance)
690?DATREG=(?DATREG AND 1)OR dir
700pulses=pd_ratio*ABS(distance)
710FOR I=1 TO pulses:PROCpulse:NEXT I
720ENDPROC
730:
740DEF PROCturn(dir,angle)
750?DATREG=(?DATREG AND 1)OR dir
760pulses=pa_ratio*angle
770FOR I=1 TO pulses:PROCpulse:NEXT I
780ENDPROC
790:
800DEF PROCpulse
810?DATREG=(?DATREG OR 8)
820?DATREG=(?DATREG AND 247)
830ENDPROC
840DEF PROCxplot
850FOR I=0 TO width STEP 4
860MOVExstart,y+I
870DRAWx,y+I
880NEXT I
890ENDPROC
900:
910DEF PROCyplot
920FOR I=0 TO width STEP 4
930MOVEx+I,ystart
940DRAWx+I,y
950NEXT I
960ENDPROC
```



Commodore 64 Listing Continued

```
2000 REM **** SCAN DIMENSIONS ****
2010 INPUT"X DIMENSION IN MM";WX
2020 IF WX/4>319 THEN 2010
2030 INPUT"Y DIMENSION IN MM";WY
2040 IF WY/4>199 THEN 2030
2050 WX=WD*INT(WX/WD):WY=WD*INT(WY/WD)
2060 RETURN
2070 :
3000 REM **** HORIZONTAL SCAN ****
3010 TG=WX:SE=LF:XS=0
3020 DR=FW:DS=DX:GOSUB7000:REM MOVE
3030 X=X+DX
3040 IF(PEEK(DATREG)AND192)=NB AND X<>TG THEN 3020
3050 GOSUB9000:REM XPLOT
3060 DX=-DX:Y=Y+DY:XS=X
3070 IF TG=0 THEN TG=WX:GOTO 3090
3080 TG=0
3090 IF Y<WY THEN WA=SE:GOSUB8000:REM NEXT STRIP
3100 IF SE=RT THEN SE=LF:GOTO 3120
3110 SE=RT
3120 IF Y<WY THEN 3020
3130 RETURN
3140 :
4000 REM **** VERTICAL SCAN ****
4010 IF X=0 THEN SE=LF:DX=WD:GOTO 4030
4020 SE=RT:DX=-WD
4030 DY=-DY:TG=0:YS=Y
4040 DR=SE:AG=30:GOSUB7100:REM TURN
4050 DR=FW:DS=DY:GOSUB7000:REM MOVE
4060 Y=Y+DY
4070 IF(PEEK(DATREG)AND192)=NB AND Y<>TG THEN 4050
4080 GOSUB9100:REM YPLOT
4090 DY=-DY:X=X+DX:YS=Y
4100 IF TG=0 THEN TG=WY:GOTO 4120
4110 TG=0
4120 WA=SE:GOSUB8000:REM NEXT STRIP
4130 IF SE=RT THEN SE=LF:GOTO 4150
4140 SE=RT
4150 IF X<WX AND X>0 THEN 4050:REM REPEAT
4160 RETURN
4170 :
5000 REM **** ENTER HIRES ****
5010 POKE 49408,1:POKE 49409,1
5020 POKE 49410,1:SYS HIRES:RETURN
5030 :
5100 REM **** LEAVE HIRES ****
5110 POKE 49408,0:POKE 49409,0
5120 POKE 49410,1:SYS HIRES:RETURN
5130 :
6000 REM **** ENTER LINESUB ****
6010 MHI=INT(X1/256):MLO=X1-256*MHI
6020 NHI=INT(X2/256):NLO=X2-256*NHI
6030 POKE 49920,MLO:POKE 49921,MHI
6040 POKE 49922,NLO:POKE 49923,NHI
6050 POKE 49924,Y1:POKE 49925,Y2
6060 SYS LINESUB:RETURN
6070 :
7000 REM **** MOVE (DR,DS) ****
7010 POKE DATREG,(PEEK(DATREG)AND 1)OR DR
7020 PL=PD*DS
7030 FOR I=1 TO PL:GOSUB7200:NEXT I
7040 RETURN
7050 :
7100 REM **** TURN (DR,AG) ****
7110 POKE DATREG,(PEEK(DATREG)AND 1)OR DR
7120 PL=PA*DS
7130 FOR I=1 TO PL:GOSUB7200:NEXT I
7140 RETURN
7150 :
7200 REM **** PULSE ****
7210 POKE DATREG,PEEK(DATREG)OR 8
7220 POKE DATREG,PEEK(DATREG)AND 247
7230 RETURN
7240 :
8000 REM **** NEXT STRIP ****
8010 DR=BW:DS=30:GOSUB7000:REM MOVE
8020 DR=WA:AG=90:GOSUB7100:REM TURN
8030 DR=FW:DS=WD:GOSUB7000:REM MOVE
8040 DR=WA:AG=90:GOSUB7100:REM TURN
8050 RETURN
9000 REM **** XPLOT ****
9010 FOR I=0 TO WD
9020 X1=XS/4:Y1=(Y+I)/4:X2=X/4:Y2=Y1
9030 GOSUB6000:REM ENTER LINESUB
9035 NEXT I
9040 RETURN
9100 REM **** YPLOT ****
9110 FOR I=0 TO WD
9120 X1=(X+I)/4:Y1=YS/4:X2=X1:Y2=Y/4
9130 GOSUB6000:REM ENTER LINESUB
9135 NEXT I
9140 RETURN
```

STARTING BLOCK

In our previous look at the BBC Micro's OS, we considered the wide range of OSBYTE calls available to the programmer, and showed how these calls enable us to control a variety of operating system routines. Here, we investigate another useful group — the OSWORD calls.

The OSBYTE calls we looked at in the previous instalment had one major limitation: they could only accept two parameters, one in the X register and one in the Y register. This is alright if you don't need to pass too many parameters to the operating system, but even with the best will in the world we can't control every routine with two parameters. OSWORD routines, however, can control more than two parameters, and are an extremely useful alternative to OSBYTE calls.

OSWORD uses an area of memory known as a *parameter block* to pass its parameters over to the operating system. This area of memory, which varies in size depending upon the OSWORD call being used, can be anywhere within the user RAM of the computer. When an OSWORD routine is called, the value in the A register tells OSWORD which of the routines is to be called, and the X and Y registers specify the address of the first byte of the parameter block. The X register holds the low byte of the address, and the Y register holds the high byte of the address. Thus, if the parameter block is situated at address &0A00, the X register will contain 00 and the Y register will hold the value &0A:

Assembler	BASIC
LDX #0	X%=0
LDY #&0A	Y%=&0A

OSWORD is called at address &FFF1, and is vectored in the same way as an OSBYTE call. However, the OSWORD vector is at address &20C. So, to call the OSWORD routine specified when the A register holds a value of 1, we would set up the parameter block with the necessary parameters, and then call the routine using the following lines of code (parameter_block is the address of the first byte of the parameter block):

```
LDX  #parameter_block MOD 256
LDY  #parameter_block DIV 256
LDA  #1
JSR  &FFF1
```

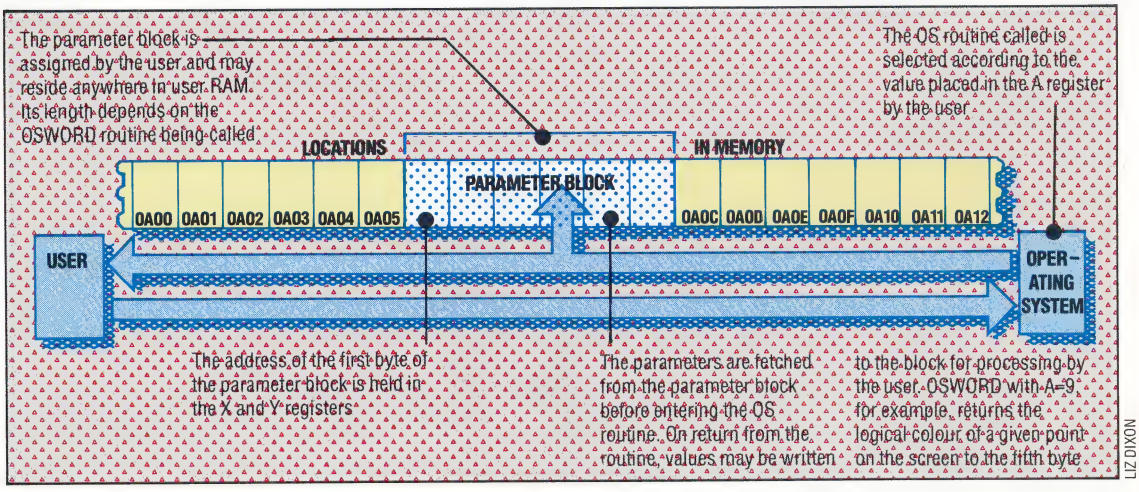
Of course, the actual values put into the parameter block depend upon the particular OSWORD call being used.

WHAT DO OSWORD CALLS DO?

Using these calls, we can read information from the current input stream into memory, perform SOUND and ENVELOPE commands, access the disk filing system, and perform many other functions. We will now examine how some of the OSWORD calls are used. To identify them, OSWORD calls are named with reference to the contents of the A register when the call to &FFF1 is made — for example, OSWORD with A=0

- *OSWORD with A=0:* This call enables us to read a line of input from the currently selected input stream — which is an extremely useful function. It allows us to specify the maximum number of characters to be accepted and the minimum and maximum values allowed for the ASCII codes of the characters being entered. The data read in from the input stream is stored in memory at the address specified in the first two bytes of the parameter block. The OSWORD parameter block is set up like this:

More Scope
An OSWORD call provides a convenient means of controlling OS routines that require more than the two parameters available through OSBYTE routines



OSWORD With A=0 Parameter Block	
Byte	**Function**
0	Low byte of buffer address
1	High byte of buffer address
2	Maximum number of characters in input
3	Minimum acceptable ASCII code
4	Maximum acceptable ASCII code

The X and Y registers point to byte 0 of the block. The following program demonstrates how this call is used; the parameter block is set up to accept up to seven characters with ASCII codes in the range 32 to 90 inclusive.

```
10 DIM C 20
20 A%=0
30 X%=C MOD 256
40 Y%=C DIV 256
50 C?0=&00
60 C?1=&0A: REM buffer at &0A00
70 C?2=7
80 C?3=32
90 C?4=90
100 CALL &FFF1
110 PRINT $(&0A00)
```

RUN the program, and type in some characters. Delete will have it's usual function, and pressing either Return or Escape will end the OSWORD routine. One important thing to note about this call is that if you try to type in characters with ASCII codes outside the specified range, they *will* be echoed to the screen, but they will not be added to the characters in the buffer memory. A 'beep' will be generated if you try to type in too many characters. In machine code, when the OSWORD call is exited, the C flag is set if the Escape key is used; otherwise C will be 0. The Y register will hold the number of characters entered, including any carriage returns.

- *OSWORDs with A=1 and A=2:* These calls allow the machine code programmer to access the BASIC variable TIME. OSWORD with A=1 allows us to read the current value of TIME. The five bytes that represent this value are then stored in the parameter block, which should, obviously, be five bytes long for this call. The least significant byte of the value is stored in byte 0 of the parameter block and the most significant byte of the result is stored in the fifth byte of the block.

OSWORD with A=2 allows us to set TIME to a particular value. The least significant byte of the value is placed in the first byte of the parameter block, the most significant byte in the fifth byte of the block, and then the call is made to OSWORD. The following program sets the value of TIME to 100:

```
10 DIM C 20
20 X%=C MOD 256
30 Y%=C DIV 256
40 A%=2
50 C?0=100
60 C?1=0
70 C?2=0
80 C?3=0
90 C?4=0
100 CALL &FFF1
```

- *OSWORDs with A=3 and A=4:* The BBC microcomputer has a second internal clock called the Interval — or Event — Timer. This is not used by TIME, but is used by the computer's 'events system', which we'll look at later in this course. This timer counts up, at one 'tick' every 1/100th of a second, from a set value, and generates an 'event' when it reaches zero. This event is easily manipulated, and can be used to trigger the computer into a sequence of operations after a certain time has expired.

OSWORD with A=3 enables us to read the current value of the interval timer into the parameter block — in a similar way to how OSWORD with A=1 reads the TIME clock. OSWORD with A=4 enables us to set this interval timer to a particular value; this is also similar to the way in which OSWORD with A=2 sets up the TIME variable. Thus, setting the parameter block to hold &FFFFFFFC would cause an event to be triggered after 0.004 seconds.

- *OSWORDs with A=5 and A=6:* These are more specialised calls, and are most useful to programmers with access to the second processor for the BBC Micro. Once one of these devices is plugged into the computer, it will communicate via the Tube interface connection and treat the computer as an I/O processor. This means that the BBC Micro acts as a slave device, and does all the donkey work for the second processor — such as handling keyboard entry, screen display and general input/output functions. These OSWORD calls enable the second processor to access memory locations within the I/O processor.

OSWORD With A = 5 And A = 6 Parameter Blocks		
Byte	**OSWORD With A = 5**	**OSWORD With A = 6**
0	LSB of I/O processor address	LSB of I/O processor address
1	Address of byte to read	Address of byte to write
2	Address of byte to read	Address of byte to write
3	MSB of I/O processor	MSB of I/O processor
4	After call, the byte at the above address	The byte to be written to the above address

The table shows the set up of the parameter blocks for A=5 (which reads a byte of I/O processor memory) and A=6 (which writes to I/O processor memory). If the address is only going to be a 16-bit address, as would be the case if the I/O processor was a standard BBC Micro, then the address is stored with its low byte at (XY+0) and its high byte at (XY+1).

- *OSWORDs with A=7 and A=8:* These two calls enable the programmer to manipulate sound in machine code programs, as well as in BASIC. OSWORD with A=7 simulates the BASIC SOUND

command, and OSWORD with A=8 is the equivalent of an ENVELOPE command. Let's consider these calls separately.

● *OSWORD with A=7:* This requires an eight-byte parameter block — the set up is shown in the table — as well as an indication of which parameter block entry corresponds to each of the parameters in the SOUND command.

OSWORD With A = 7 Parameter Block	
Byte	**Function**
0	LSB of CHANNEL NUMBER
1	MSB of CHANNEL NUMBER
2	LSB of AMPLITUDE
3	MSB of AMPLITUDE
4	LSB of PITCH
5	MSB of PITCH
6	LSB of DURATION
7	MSB of DURATION

The following program shows the call in action; it simulates the SOUND 1,-15,100,30 command.

```
10  DIM C 20
20  X%=C MOD 256
30  Y%=C DIV 256
40  A%=7
50  C?0=1
60  C?1=0:REM Channel 1
70  C?2=-15 MOD 256
80  C?3=-15 DIV 256
90  C?4=100
100 C?5=0
110 C?6=30
115 C?7=0
120 CALL &FFF1
130 END
```

Obviously, the same effect is a lot easier to achieve using the SOUND command, and so this call is of minimal use in BASIC.

● *OSWORD with A=8:* This call is even more long-winded than OSWORD with A=7 — the parameter block is 14 bytes long! As usual, the X and Y registers are used to point to the parameter block. The first entry in the parameter block holds the ENVELOPE number, and the remaining thirteen bytes of the block hold the other ENVELOPE parameters. You'll find it much easier to use ENVELOPE in BASIC to set your sound envelopes up.

OSWORD GRAPHICS CALLS

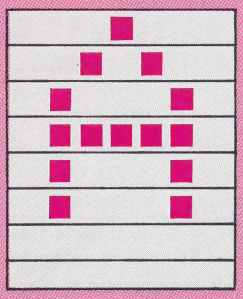
● *OSWORD with A = 9:* This performs the equivalent of a BASIC POINT command from within a machine code program. It returns the logical colour at a given point, or the value 255 if the point specified is not on the screen. The value is returned in the fifth byte of the parameter block, set up like this:

OSWORD With A = 9 Parameter Block	
Byte	**Function**
0	LSB of X co-ordinate
1	MSB of X co-ordinate
2	LSB of Y co-ordinate
3	MSB of Y co-ordinate
4	Logical colour

OSWORD With A = 10

This is quite a useful call if you want to print large characters to the screen. It enables you to get the actual character definition of any Mode 0 to 6 character. You need a nine-byte parameter block, pointed to by the X and Y registers, as usual. The first byte of the block should contain the ASCII code of the character

OSWORD With A = 10 Parameter Block	
Byte	**Function**
0	Character required
1	First line of definition
2	Second line
3	Third Line
4	Fourth line
5	Fifth line
6	Sixth line
7	Seventh line
8	Last line of definition



The values returned in the parameter block represent a binary representation of the character definition in which 1 is an illuminated pixel on the screen and 0 is an unlit pixel

● *OSWORD with A=11:* This call lets us inspect the colours specified in the VDU 19 set of commands; it enables us to find out what *physical* colour is associated with a particular logical colour. This can be useful if you've forgotten what colours you've specified on a particular VDU 19 call. It requires a five-byte parameter block, and when the call has been made the first byte of the parameter block should hold the value of the logical colour of interest. As an example of its use, let's assume that you have already executed a VDU19,2,4,0,0,0 command. OSWORD with A = 11 and a value of 2 in the first byte of the parameter block would return the values shown in the margin.

Byte	Value
0	2
1	4
2	0
3	0
4	0

● *OSWORD with A = 12:* This is a faster version of VDU 19.

● *OSWORD with A = 13:* Another call that seems to have few obvious uses. It returns the X and Y co-ordinates of the last two points visited by the graphics cursor, and can come in handy at times.

From our brief survey, you can see that the OSWORD calls are very versatile and make many facilities of the BBC Micro easily available.

DATABASE

National Replacement Character Sets

This appendix describes the 10 National Replacement Character Set (NRCS) translation tables you can choose on the Configuration settings sheet under Other File-Translation and the Communications settings sheet under Terminal Translation. National replacement character sets are in widespread use in Europe and are national standards in many countries. An NRCS translation table is created by taking the LICS character set and replacing some characters (typically #, @, {, }, [,], \, |, ‘, ~, ^, and _) with international characters, such as é or à.

The NRCS translation table you select on the Communications settings sheet is active only during terminal emulation. If you attempt to enter a character that is not in the NRCS you select, Symphony displays its **unknown character** symbol (■). This includes the LICS characters replaced by the international characters, and all characters that use the eighth bit (LICS codes 128 through 255). If you attempt to transmit a character that is not in the NRCS you select, Symphony substitutes a character code 63, which causes a question mark (?) to appear at the receiving end.

The NRCS translation table you select on the Configuration settings sheet is active only during printing to or importing from a print file. The 10 available translation tables are shown in the table below.

Language	LICS Character	NRCS Character	Description
Spanish			
	#	£	Pound sign
	@	§	Section sign
	[	¡	Inverted exclamation mark
	\	Ñ	Uppercase N with tilde
	]	¿	Inverted question mark
	{	°	Degree sign
	¦	ñ	Lowercase n with tilde
	}	ç	Lowercase c with cedilla
British			
	#	£	Pound sign
French-Canadian			
	@	à	Lowercase a with grave accent
	[	â	Lowercase a with circumflex
	\	ç	Lowercase c with cedilla
	]	ê	Lowercase e with circumflex
	^	î	Lowercase i with circumflex
	`	ô	Lowercase o with circumflex
	{	é	Lowercase e with acute accent
	¦	ù	Lowercase u with grave accent
	}	è	Lowercase e with grave accent
	~	û	Lowercase u with circumflex
Danish/Norwegian			
	@	Å	Uppercase A with umlaut
	[	Æ	Uppercase AE ligature
	\	Ø	Uppercase O with slash
	]	Å	Uppercase A with ring
	^	Ü	Uppercase U with umlaut
	`	ä	Lowercase a with umlaut
	{	æ	Lowercase ae ligature
	¦	ø	Lowercase o with slash
	}	å	Lowercase a with ring
	~	ü	Lowercase u with umlaut
Finnish			
	[	Ä	Uppercase A with umlaut
	\	Ö	Uppercase O with umlaut
	]	Å	Uppercase A with ring
	^	Ü	Uppercase U with umlaut
	`	é	Lowercase e with acute accent
	{	ä	Lowercase a with umlaut
	¦	ö	Lowercase o with umlaut
	}	å	Lowercase a with ring
	~	ü	Lowercase u with umlaut
French			
	#	£	Pound sign
	@	à	Lowercase a with grave accent
	[	°	Degree sign
	\	ç	Lowercase c with cedilla
	]	§	Section sign
	{	é	Lowercase e with acute accent
	¦	ù	Lowercase u with grave accent
	}	è	Lowercase e with grave accent
	~	¨	Umlaut

Language	LICS Character	NRCS Character	Description
German			
	@	§	Section sign
	[	Ä	Uppercase A with umlaut
	\	Ö	Uppercase O with umlaut
	]	Ü	Uppercase U with umlaut
	{	ä	Lowercase a with umlaut
	¦	ö	Lowercase o with umlaut
	}	ü	Lowercase u with umlaut
	~	ß	Lowercase German sharp s
Swedish			
	@	É	Uppercase E with acute accent
	[	Ä	Uppercase A with umlaut
	\	Ö	Uppercase O with umlaut
	]	Å	Uppercase A with ring
	^	Ü	Uppercase U with umlaut
	`	é	Lowercase e with acute accent
	{	ä	Lowercase a with umlaut
	¦	ö	Lowercase o with umlaut
	}	å	Lowercase a with ring
	~	ü	Lowercase u with umlaut
Italian			
	#	£	Pound sign
	@	§	Section sign
	[	°	Degree sign
	\	ç	Lowercase c with cedilla
	]	é	Lowercase e with acute accent
	`	ù	Lowercase u with grave accent
	{	à	Lowercase a with grave accent
	¦	ò	Lowercase o with grave accent
	}	è	Lowercase e with grave accent
	~	ì	Lowercase i with grave accent
Swiss			
	#	ù	Lowercase u with grave accent
	@	à	Lowercase a with grave accent
	[	é	Lowercase e with acute accent
	\	ç	Lowercase c with cedilla
	]	ê	Lowercase e with circumflex
	^	î	Lowercase i with circumflex
	`	ô	Lowercase o with circumflex
	_	è	Lowercase e with grave accent
	{	ä	Lowercase a with umlaut
	¦	ö	Lowercase o with umlaut
	}	ü	Lowercase u with umlaut
	~	û	Lowercase u with circumflex